المملكـة العربيــة الســعودية

المؤسسة العامة للتعليم الفني والتدريب المهني

الإدارة العامة لتصميم وتطوير المناهج



دعم فني

برمجة الحاسب

١٤١ حاب



الحمد لله وحده، والصلاة والسلام على من لا نبي بعده، محمد وعلى آله وصحبه، وبعد:

تسعى المؤسسة العامة للتعليم الفني والتدريب المهني لتأهيل الكوادر الوطنية المدربة القادرة على شغل الوظائف التقنية والفنية والمهنية المتوفرة في سوق العمل، ويأتي هذا الاهتمام نتيجة للتوجهات السديدة من لدن قادة هذا الوطن التي تصب في مجملها نحو إيجاد وطن متكامل يعتمد ذاتياً على موارده وعلى قوة شبابه المسلح بالعلم والإيمان من أجل الاستمرار قدماً في دفع عجلة التقدم التنموي: لتصل بعون الله تعالى لمصاف الدول المتقدمة صناعياً.

وقد خطت الإدارة العامة لتصميم وتطوير المناهج خطوة إيجابية تتفق مع التجارب الدولية المتقدمة في بناء البرامج التدريبية، وفق أساليب علمية حديثة تحاكي متطلبات سوق العمل بكافة تخصصاته لتلبي متطلباته ، وقد تمثلت هذه الخطوة في مشروع إعداد المعايير المهنية الوطنية الذي يمثل الركيزة الأساسية في بناء البرامج التدريبية، إذ تعتمد المعايير في بنائها على تشكيل لجان تخصصية تمثل سوق العمل والمؤسسة العامة للتعليم الفني والتدريب المهني بحيث تتوافق الرؤية العلمية مع الواقع العملي الذي تفرضه متطلبات سوق العمل، لتخرج هذه اللجان في النهاية بنظرة متكاملة لبرنامج تدريبي أكثر التصاقاً بسوق العمل، وأكثر واقعية في تحقيق متطلباته الأساسية.

وتتناول هذه الحقيبة التدريبية " برمجة الحاسب " لمتدربي قسم" دعم فني " للكليات التقنية موضوعات حيوية تتناول كيفية اكتساب المهارات اللازمة لهذا التخصص.

والإدارة العامة لتصميم وتطوير المناهج وهي تضع بين يديك هذه الحقيبة التدريبية تأمل من الله عز وجل أن تسهم بشكل مباشر في تأصيل المهارات الضرورية اللازمة، بأسلوب مبسط يخلو من التعقيد، وبالاستعانة بالتطبيقات والأشكال التي تدعم عملية اكتساب هذه المهارات.

والله نسأل أن يوفق القائمين على إعدادها والمستفيدين منها لما يحبه ويرضاه: إنه سميع مجيب الدعاء.

الإدارة العامة لتصميم وتطوير المناهج



برمجة الحاسب

مقدمة و حل المشكلة

مقدمة و حل المشكلة

١

تمهيد

من المعلوم اليوم أن الحاسبات انتشرت انتشارا واسعا وكبيرا لدرجة أنها أصبحت في كل موقع وفي كل مكان ولا يمكن الاستغناء عنها بأي حال من الأحوال، وذلك لما تقوم به من أعمال كبيرة وعظيمة و لما تتمتع به من قدرة عالية على إجراء العمليات الحسابية وغيرها من العمليات في وقت قصير جدا ، كما أنها تتميز بالقدرات العالية على معالجة الكم الهائل من البيانات حفظا وترتيبا واسترجاعا وبحثا وغيرها الكثير من العمليات.

ونظرا لما سبق أصبح لزاما علينا - لكي نواكب هذا العصر ولكي ننهض بوطننا وشعبنا و أمتنا - أن نعرف الكثير عن هذه الحاسبات وكيف يمكن التعامل معها والاستفادة منها. ومن الوسائل التي تساعدنا على الاستفادة من هذه الحاسبات معرفة وإتقان إحدى لغات البرمجة المعروفة والمشهورة هذه الأيام، ومن هذه اللغات المشهورة والتي بدأت تستخدم على نطاق واسع لغة الجافا Java language وذلك لما تتمتع به من قدرة على العمل (التنفيذ) مع كل الحاسبات وسهولة كتابة البرامج المختلفة سواء منها البسيطة أو الكبيرة.

وهذه الحقيبة تقدم شرحا تفصيليا للمفردات الأساسية المكونة للغة الجافا وكذلك كتابة بعض البرامج البسيطة والمتوسطة باستخدام هذه اللغة. ففي الوحدة الأولى مقدمة للغات البرمجة المختلفة وشرح لكيفية تحليل وحل المشاكل البسيطة باستخدام خرائط التدفق والكود الزائف وكذلك كتابة البرامج البسيطة ومعرفة المفردات الأساسية للغة من متغيرات وأنواع البيانات والعمليات الحسابية والمنطقية وغيرها من العمليات تم شرحها وتوضيحها في الوحدة الثانية. أما الوحدة الثالثة فإنها تتناول الحلقات (looping) بأنواعها المختلفة والتفريعات (branching) وكيفية كتابتها والاستفادة منها في حل البرامج البسيطة والمتوسطة، بالإضافة إلى تنفيذ هذه البرامج على الحاسب.

الوحدة الأولى

مقدمة وحل المشكلة

في هذه الوحدة نعرض مقدمة عن ماهية برنامج الحاسب ولغة البرمجة وأنواع لغات البرمجة وأهمية مهنة البرمجة، ثم بعد ذلك نشرح القواعد التي تساعد في تحليل المشكلة ومعرفة عناصرها المكونة لها و كيف يمكن تجزئة المشكلة إلى أجزاء صغيرة يسهل التعامل معها، وفيها أيضا نوضح رموز رسم خرائط التدفق ثم رسم هذه الخرائط للمشكلة بعد كتابة الخوارزم والتي تعطي صورة لحل المشكلة.

الفصل الأول: مقدمة

الجدارة:

معرفة ماهية برنامج الحاسب ولغات البرمجة وأنواعها

الأهداف:

عندما تكمل هذه الوحدة يكون لديك القدرة على:

١ - فهم ماهية برنامج الحاسب
٢ - معرفة لغات البرمجة المختلفة التي يمكن أن يراها
٣ - الإخبار بأهمية مهنة البرمجة
٤ - معرفة ما هو علم صناعة البرمجيات

مستوى الأداء المطلوب

أن يصل المتدرب إلى إتقان هذه الجدارة بنسبة ١٠٠٪

الوقت المتوقع للتدريب: ساعة واحدة

الوسائل المساعدة:

- قلم
- دفتر

متطلبات الجدارة:

اجتياز جميع الحقائب السابقة

الفصل الأول: مقدمة

نظراً للتطور الكبير في تقنية صناعات الحاسبات الآلية وانتشارها في جميع مجالات الحياة المختلفة، واستخداماتها المتعددة في شتى المجالات، فإنه أصبح لزاماً علينا معرفة هذه الحاسبات وكيفية التعامل معها والاستفادة منها لأنها توفر الجهد والوقت وتنجز كثير من الأعمال بدقه كبيرة بالإضافة إلى قدراتها الكبيرة في الاحتفاظ بالبيانات. ومن الطرق الشائعة للاستفادة من القدرات الكبيرة للحاسبات هو: بناء البرامج التي تقوم بحل كثير من المشكلات توفيراً للجهد والوقت ووصولاً إلى الدقة المطلوبة، وفي هذه الوحدة سوف نلقي الضوء على ماهية برنامج الحاسب وكذلك أنواع لغات البرمجة المختلفة. ثم بعد ذلك نبين أهمية مهنة البرمجة وصناعة البرمجيات.

برنامج الحاسب

البرنامج هو عبارة عن مجموعة من التعليمات تعطى للحاسب للقيام بعمل ما مثل حساب مجموع قيم مختلفة، حساب المتوسط الحسابي، حساب مضروب عدد معين ....الخ

والبرنامج هو الذي يحدد للحاسب كيفية التعامل مع البيانات للحصول على النتائج المطلوبة. والبرنامج يكتب بواسطة المبرمج (Computer Programmer) الذي يفهم المشكلة ويقترح الحل وينفذه لحل هذه المشكلة ويجب أن يكون البرنامج في مجموعه صحيحاً وواضحا وليس فيه لبس أو غموض.

والبرمجيات (Software) هي التي تسهل للمستخدم استخدام المكونات المادية (Hardware) بكفاءة وراحة ويمكن تقسيم البرمجيات إلى ثلاثة أنواع رئيسية وهي: -

١ - برامج التشغيل Operating System

مثل النوافذ(windows) و Dos، Unix، Linux، VMS وغيرها. وهي عبارة عن برامج تقوم بدور الوسيط بين المستخدم والمكونات المادية وهي تمكن المستخدم من استخدام المكونات المادية للحاسب بكفاءة وبراحة، كما أنها تساعد المستخدم في إنشاء نظام الملفات وغيرها. ومن برامج التشغيل ما يصلح للعمل في الشبكات مثل Unix، Windows، ومنها الذي يستخدم مع الحاسب فقط مثل Dos.

٢ - **برامج التطبيقات** Application Programs

وهي برامج تساعد في إنشاء كثير من التطبيقات مثل إنشاء قاعدة بيانات والرسم باستخدام الحاسب و غيرها ومن أمثلة هذه البرامج: -

برنامج الأوتوكاد Autocad– الاكسيل Excel - الأكسس Access - الأوراكلOracle - الفوتوشوب Fotoshop وغيرها كثير.

٣ - **لغات البرمجة** Programming Languages

وهذه اللغات هي التي تستخدم في بناء البرامج المختلفة وهي تتراوح من اللغات التي تتعامل مباشرة مع المكونات المادية للحاسب والأخرى التي تتطلب تحويلها من صورتها التي تكتب بها إلى صورة أخرى يستطيع الحاسب التعامل معها.

ويوجد العديد من لغات البرمجة المستخدمة اليوم وهذه اللغات يمكن تقسيمها إلى ثلاث أنواع رئيسية هي: -

١ - لغة الآلة **Machine languages**

٢ - لغات التجميع **Assembly languages**

٣ - لغات المستوى العالي **High level languages**

لغة الآلة Machine Language

وهي اللغة الوحيدة التي يفهمها الحاسب ويستطيع التعامل معها. وهذه اللغة تعتبر لغة خاصة لكل حاسب وقد تختلف من حاسب إلى آخر وهي تعتمد على المكونات المادية للحاسب نفسه، ولغة الآلة تتكون من مجموعة أرقام من بين 0، 1 التي تعطي تعليمات للحاسب للقيام بمعظم العمليات الأساسية واحدة بعد الأخرى، وهي تختلف من حاسب إلى حاسب آخر ولذلك فإننا نجد أن نفس البرنامج الذي يعمل على حاسب معين قد لا يعمل على حاسب آخر يختلف عنه في المكونات المادية. و لغة الآلة من اللغات الصعبة في التعلم للإنسان حتى بالنسبة للمبرمجين لأنها عبارة عن مجموعة من الأرقام (٠، ١) فقط. وللتغلب على هذه الصعوبة تم اقتراح لغة أخرى تعتمد على استخدام اختصارات معبرة من اللغة الإنجليزية للتعبير عن العمليات الأولية التي يقوم بها الحاسب وهذه اللغة هي لغة التجميع.

لغة التجميع Assembly Languages

هي لغة تستخدم اختصارات معبرة من اللغة الإنجليزية لتعبر بها عن العمليات الأولية التي يقوم بها الحاسب مثل إضافة Add و حفظ Store وطرح Sub وغيرها.

مثال على ذلك

```
Load    A
Add     B
Store   C
```

ونظراً لأن هذه اللغة تستخدم كلمات مختصرة من اللغة الإنجليزية فإنها تحتاج محولاً لكي يحولها إلى لغة الآلة وهو ما يسمى المجمع assembler الذي يقوم بتحويل لغة التجميع إلى لغة الآلة كي يفهمها الحاسب ويستطيع تنفيذها ، وبالرغم من تقليل المجهود الملقى على عاتق المبرمج للقيام بعملية البرمجة إلا أنه ما زالت توجد مشقة عند حل أبسط المسائل لأن ذلك يتطلب معرفة وكتابة العديد من التعليمات، وهذا ما دفع المبرمجين للتفكير في لغات أخرى تقلل المجهود الكبير اللازم لكتابة الكثير من التعليمات فكانت لغات البرمجة ذات المستوى العالي.

لغات البرمجة ذات المستوى العالي High Level Languages

وهذه اللغات كتبت بحيث تستخدم بعض الكلمات الإنجليزية العادية بنفس معانيها حيث يقوم كل أمر منها بتنفيذ العديد من الواجبات، وهذه اللغات كسابقتها تحتاج إلى مترجمات Compilers التي تقوم بتحويل التعليمات(الأوامر) إلى لغة الآلة، .وهذه اللغات تستخدم العلاقات والعوامل الرياضية المتعارف عليها. مثال ذلك

$$Sum = A + B + C$$

وهذه اللغات تعتبر سهلة ومرغوبة من وجهة نظر المبرمجين بالمقارنة بلغات التجميع ولغة الآلة وذلك لسهولة كتابتها وفهمها وحل المشاكل باستخدامها ، ومن أمثلة هذه اللغات لغة C ، ++C، الباسكال Pascal، الفورترانFortran، البيسك Basic، الآداADA ،الجافا Java وغيرها.

ومن المعلوم أن عملية تحويل البرنامج من لغة ذات مستوى عال إلى لغة الآلة تستهلك وقتاً ولذلك تم تطوير نسخ من لغات المستوى العالي بحيث تستخدم برنامج مفسرInterpreter والذي يقوم بترجمة الكود سطرا سطرا أثناء التنفيذ.

.وبالرغم من أن البرامج المترجمة الناتجة من عملية الترجمة باستخدام المترجم compiler تكون أسرع في التنفيذ عن البرامج التي تستخدم المفسر (Interpreter) إلا أنه يفضل وجود نسخة من اللغة تعمل باستخدام المفسر وذلك لسهولة التغيير والحذف والإضافة والتصحيح. وبعد الانتهاء من كل التعديلات والوصول إلى نسخة نهائية فإنه يتم استخدام المترجم لترجمة البرنامج وإنتاج نسخة تنفيذية حتى تكون أسرع في التنفيذ بعد ذلك عند تشغيلها على الحاسب.

أهمية مهنة البرمجة

من المعلوم أن الذي يقوم بكتابة البرامج لحل المشكلات الكثيرة والمعقدة هم المبرمجون ولا يمكن الاستغناء عنهم بحال من الأحوال لأن دورهم مهم وحيوي وتكثر الحاجة لهم في شتى المجالات وذلك لعمل الآتي: -

١ - كتابة برامج وبناء الأنظمة المختلفة لحل المشاكل وتبسيط التعامل مع الحاسب.

٢ - المسئولية الكاملة عن إصلاح ما يحدث من أعطال أو حل المشاكل التي تحدث في الأنظمة المختلفة.

٣ - بناء واجهة المستخدم المختلفة في كثير من اللغات والتطبيقات.

٤ - بناء نظم التشغيل المختلفة مثل Unix، Windows وغيرها من النظم. فمثلاً تستخدم لغة C في بناء نظام التشغيل Unix.

٥ - برامج المواجهة المختلفة في الأنظمة المختلطة الرقمية و التماثلية.

صناعة البرمجيات

تعتبر صناعة البرمجيات في عصرنا الحالي من الصناعات المهمة جدا والتي تتطور باستمرار نتيجة التطور الهائل في صناعة الحاسبات الآلية، ولذلك فإن هذه الصناعة تتطلب مبرمجين مهرة ولديهم القدرة على تحليل وحل المشاكل بالإضافة إلى إلمام بكل المستجدات والعلوم والتطوير المتعلق بالحاسب وصناعة الحاسبات و ذلك حتى يستطيعوا مواكبة تطوير البرامج والنظم المختلفة للاستفادة العظمى من التقدم في الحاسبات.

تمارين

1- أكمل العبارات الآتية بكلمات مناسبة

أ - من أمثلة برامج التشغيل.................، ....، ....

ب - تُقَسَّم البرمجيات إلى ثلاثة أنواع رئيسية هي: -

١ - ....

٢ - ....

٣ - ....

ج - يوجد العديد من لغات الحاسب العالية المستوى مثل...............، ....، ....، ....

د - برنامج الحاسب هو عبارة عن .... تُعطَى للحاسب للقيام بعمل ما مثل............، ....

2- ضع علامة (√) أمام العبارة الصحيحة وعلامة (×) أمام العبارات الخاطئة

أ- برامج التشغيل تقوم بدور الوسيط بين المكونات المادية المكونة للحاسب والمسُتخدم ( )

ب- لغة الآلة تعتبر أسهل لغات البرمجة ( )

ت- البرامج المكتوبة بلغة المستوى العالي يتم تنفيذها مباشرةً ( )

ث- الحاسب لا يفهم إلا لغة الآلة ( )

ج- المترجمات تقوم بتحويل لغة البرمجة إلى لغة الآلة ( )

الفصل الثاني

حل المشكلة Problem Solving

الجدارة:

المساعدة في تحليل المشكلة وتخطيط الحل لهذه المشكلة باستخدام خرائط التدفق والخوارزميات

الأهداف:

عندما تكمل هذه الوحدة يكون لديك القدرة على

1- معرفة أجزاء المشكلة الرئيسية والفرعية
2- تحديد الاحتياجات المطلوبة لحل المشكلة
3- المشاركة بوضع ورسم خريطة التدفق للبرنامج
4- المشاركة في كتابة خوارزمية الحل للمشكلة

مستوى الأداء المطلوب:

أن يصل المتدرب إلى إتقان هذه الجدارة بنسبة 100٪

الوقت المتوقع للتدريب: 8 ساعات

الوسائل المساعدة:

- قلم
- دفتر

متطلبات الجدارة:

اجتياز جميع الحقائب السابقة

الفصل الثاني

حل المشكلة

Problem Solving

مقدمة

القدرة على حل المشاكل بواسطة البرمجة هي مهارة وطريقة مرتبة ولا تعتمد على العشوائية، وهذه القدرة يمكن اكتسابها وتعلمها باتباع بعض القواعد التي تساعد على ذلك، وبعض هذه القواعد ذكرها رين ديكارت الرياضي والفيلسوف المعروف وهي: -

١ - لا يمكن قبول أي شيء حقيقة مسلمة إلا إذا ثبت ذلك بالتجربة والمشاهدة.

٢ - كل مشكلة أو معضلة يتم تبسيطها وتقسيمها إلى أجزاء عدة كلما أمكن ذلك.

٣ - فكر بطريقة منظمة ومنطقية وذلك بالبدء بالأجزاء البسيطة والسهلة الفهم ثم التدرج إلى الأجزاء الأصعب وهكذا حتى يتم الانتهاء من المشكلة.

٤ - المراجعة لجميع الأجزاء حتى يكتمل الحل.

وبالرغم من أن هذه القواعد تم وضعها قبل ٣٠٠عام من صناعة أول حاسب إلكتروني إلا أنها ما زالت مطبقة وصالحة للاستخدام، والتفكير الجيد والمنظم لتعريف وتحديد المشكلة ضروري ومهم جداً وأساسي للحصول على نتائج صحيحة وبخاصة عند التعامل مع الحاسب، ولذلك فإن أول خطوة لحل المشكلة هو فهمها.

فهم المشكلة

المشاكل دائماً تظهر أكثر تعقيداً عن الحقيقة التي هي عليها وذلك لعدم فهم المشكلة. ومن معالجة القاعدة الأولي لديكارت والتي تنص على التأكد مما تريد يمكن الحصول على القاعدة الأولى لحل المشكلة وهي: -

قاعدة ١

حلل المشكلة بعناية فائقة محاولاً فهم كل جزئياتها وتحديد كل المتطلبات للحصول على الحل المقبول وفهم كل ما يؤدي للحصول على الحل المقبول للمشكلة.

فإذا وجد حل، بين كيف يمكن العمل لتحقيق هذا الحل. ولذا يجب تحديد مستوى النتائج المطلوبة في المراحل الأولى كما يجب أن تكون الأهداف واضحة ومعلومة وكذلك الوسائل اللازمة لتحقيق هذه

الأهداف ، وملخص هذه القاعدة هو أن فهم المشكلة يمثل نصف الحل وكذلك الفهم الجيد والصحيح والكامل للمشكلة يعطي دائماً نتائج واضحة وصحيح.

تقسيم المشكلة

بزيادة فهم المشكلة يزداد تبعاً له وضوح تفصيلات وأبعاد المشكلة، وبالتالي تصبح المشكلة أكثر تفصيلاً وثباتاً ووضوحا، مما يجعل من الصعب التعامل مع كل هذه التفاصيل في نفس الوقت، وهذا يوضح القاعدة الثانية لديكارت والتي تنص على : -

قاعدة٢

"حاول أن تقسم المشكلة إلى أجزاء بسيطة وغير معتمدة على بعضها البعض ثم ركز على كل جزء على حدة" . وفي هذا الإطار يمكن استخدام العديد من الطرق المختلفة لتقسيم المشكلة ، وبذلك يمكن الحصول على القواعد الفرعية التالية من القاعدة الثانية

قاعدة ٢أ

حاول تقسيم المشكلة إلى مجموعة مشاكل (أجزاء) بسيطة متتابعة، وحتى نحصل على الحل الكامل للمشكلة الأصلية بحل المشاكل الفرعية البسيطة الواحدة تلو الأخرى. والغرض من تقسيم المشكلة هو العمل مع جزء واحد فقط وعزل تأثير الأجزاء الأخرى حتى يسهل التعامل معه، ولكن يجب عدم إهمال ما تقوم به الأجزاء الأخرى من المشكلة لأنه لا يمكن أن تكون معزولة نهائياً عن باقي الأجزاء، ومن المؤكد أن بعض أجزاء المشكلة يجب أن ينظر له ويتم التعامل معه أولاً لأن الأجزاء الأخرى تتأثر به أو تعتمد على النتائج التي تنتج منه. وعند حل كثير من المشاكل فإن ذلك يتضمن تكرار التعامل مع بعض الحالات والأوضاع مثل المستهلكين ، نتائج التجارب......الخ، وفي مثل هذه المشاكل (الحالات) يجب التأكيد على كيفية التعامل مع الحالات الفردية. وإذا كان حل أحد هذه المشاكل (المسائل) كافياً وصحيحاً يمكن للمبرمج أن يعيد استخدام هذا الحل لكل المشاكل المشابهة في جميع الحالات.

قاعدة٢ب

إذا كانت المشكلة تتضمن بعض العمليات التي يعاد تكرارها حاول عزل العمليات التي لا تتطلب الإعادة من تلك التي تتطلب الإعادة.

إذا كنت لا تستطيع أن تقرر من أين تبدأ فإن هذا يحدث لوجود بعض الحالات الخاصة التي تسبب إزعاجاً عند فصلها. وفي هذه الحالة يكون من المفيد أن يتم إهمال هذه الحالات الخاصة وكذلك

الحالات غير المفيدة وغير النافعة في البداية ثم في نهاية الحل يمكن التعامل مع جميع الحالات بما فيها الحالات الخاصة وذلك بعد إجراء بعض التعديلات البسيطة على الحل المقترح.

قاعدة٢ج

في البداية حاول إيجاد حل للمشاكل في الحالات البسيطة أو الحالات المشهورة وعند الوصول إلى حل مرضٍ وصحيح يمكن تطوير هذا الحل ليشمل الحالات الخاصة والمعقدة.

ومن هذه القاعدة نستنتج أن التعامل مع الحالات البسيطة والمشهورة وعند الحصول منها على نتائج مرضية فإن ذلك يشجع على إمكانية الوصول إلى حل للحالات الخاصة. وأما إذا لم نستطيع الحصول على نتائج في الحالات البسيطة فلن نستطيع الحصول على نتائج صحيحة في الحالات الخاصة والمعقدة. ونلخص ذلك بأن تبدأ بالتعامل مع الأجزاء البسيطة ثم تتدرج إلى الأصعب فالأصعب وهكذا.

عملية حل المشاكل

القواعد المؤدية للحل يمكن أن تطبق بطرق مختلفة، كما أنها يجب أن تطبق ببطء وعناية وهذا ما توضحه القاعدة الثالثة

قاعدة٣

"عند تقسيم المشكلة إلى أقسام صغيرة يجب أن يكون التقسيم على خطوات متعددة بحيث تستخدم القواعد العامة في المراحل الأولى ثم يتم الانتقال إلى المراحل الخاصة بعد ذلك"

المراحل الأولى في الحل تتطلب اعتبارات عامة وواسعة بينما المراحل المتأخرة تتطلب التركيز على التفاصيل والانتقال من العام إلى الخاص وهذا ما يعرف بطريقة من الأعلى إلى الأسفل top-down design. ويقترح ألا يتجاوز عدد الأجزاء المقسمة في كل خطوة ٥ أجزاء. والقاعدة الأساسية في عملية التقسيم هي أن يستمر التقسيم حتى يمكن عزل الأجزاء عن بعضها البعض، وأن يكون حل هذه الأجزاء سهلاً. والقدرة على التقسيم تتطلب مهارة عالية وخبرة إلا أن هذه الخبرة يمكن اكتسابها وتطويرها وتنميتها.

قاعدة ٤

"في كل مرحلة من المراحل يجب مراجعة الحل المقترح ليتم التأكد من أنه كامل وصحيح"

يعني ذلك أن مراجعة واحدة للحل لن تكون كافية ويجب تطبيق القاعدة الرابعة عند كل مرحلة. بعد حل واحد من البرامج الفرعية أو الأجزاء يجب إعادة النظر في الحل المقترح لنرى إذا كان يحقق المطلوب

بدقة من هذا البرنامج الفرعي ، وعند تجميع حلول البرامج الفرعية يجب التأكد من التوافق بين كل هذه الحلول للبرامج الفرعية والتأكد من أنها تحقق المطلوب وأنها تأخذ في الحسبان كل الحالات الخاصة. وأخيراً لا تتردد في مراجعة الحلول المقترحة فإنك سوف تجد شيئاً ما يجب أن يضاف أو يعدل أو يحذف.............الخ.

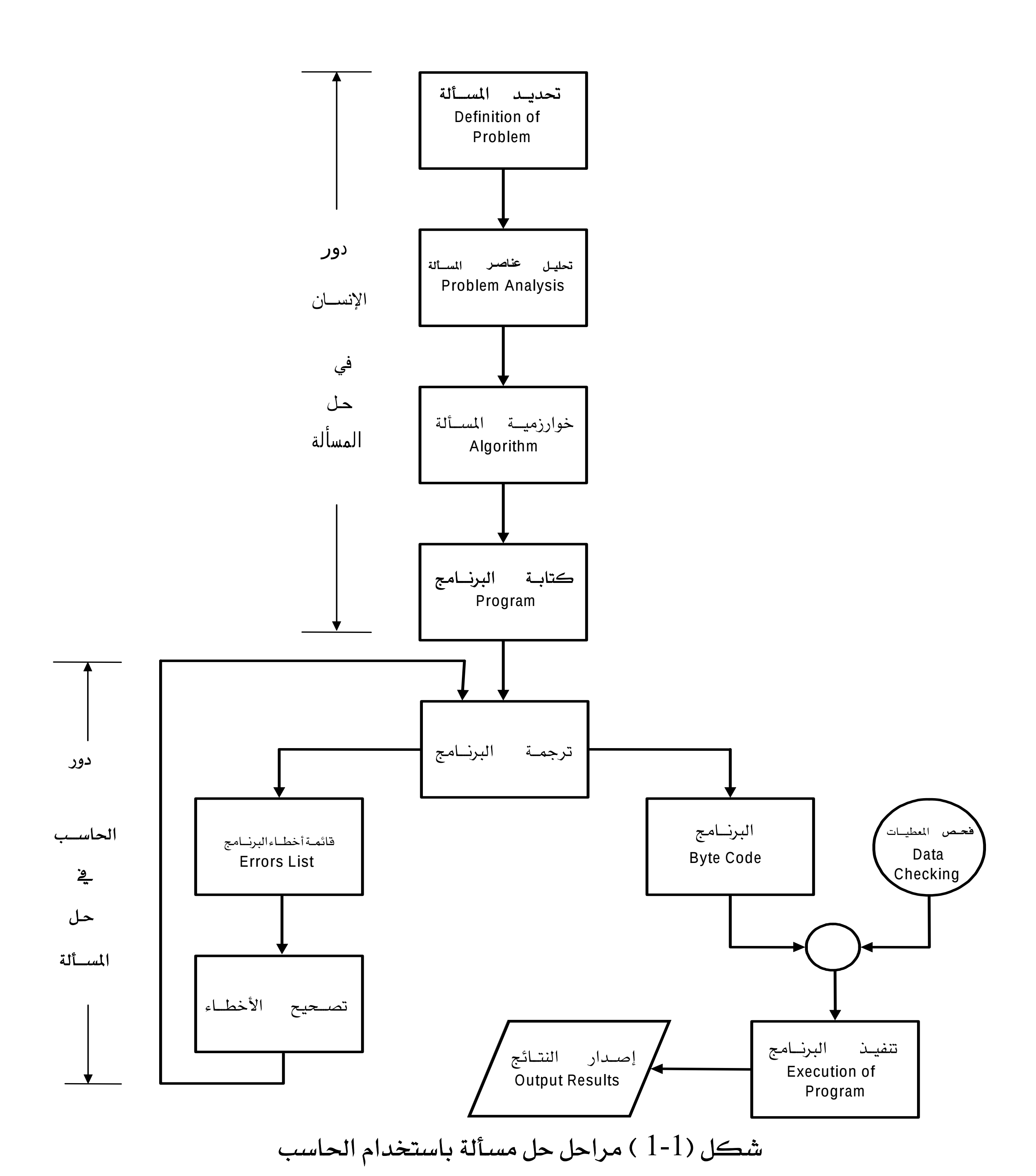
الخوارزم والكود الزائف Algorithm and Pseudo Code

بعد أن استعرضنا خطوات التفكير لحل أية مسألة برمجية وقبل أن ندخل في تفاصيل كتابة الخوارزم لحل المسألة نقول أن الحل يمر بمرحلتين كما هو مبين بشكل (١-1).

المرحلة الأولى

هذه المرحلة تمثل دور الإنسان في حل المسألة وتتكون من عدة خطوات تعرضنا لها فيما سبق ونجملها فيما يأتي:

- تحديد معالم المسألة
- تحليل عناصرها، وذلك بمعرفة معطياتها، والهدف الأساسي لها، وأهم النتائج المطلوبة منها، وما هي الصورة المراد عرض النتائج فيها، وكذلك صورة تقديم المعطيات.
- البحث والتفكير في طريقة حل المسألة
- تدوين الحل في خطوات متسلسلة متعاقبة، يعبر عنها باللغة العادية محكومة بالمنطق الرياضي. هذه الخطوات في مجموعها تسمي بالخوارزم Algorithm، كما يمكن تمثيل هذه الخطوات والارتباط فيما بينها بما يعرف بخريطة التدفق flowchart ، وذلك لكي تساعد في تسلسل المنطق العام لحل المسألة – وسوف نتعرض بالتفصيل لشرح كل من الخوار زم وخرائط التدفق لاحقا في هذا الفصل..
- كتابة البرنامج



شكل (1-1) مراحل حل مسألة باستخدام الحاسب

المرحلة الثانية

وهذه المرحلة تمثل دور الحاسب نفسه في حل المسألة، والتي تبدأ بترجمة البرنامج المكتوب بلغة المستوى العالي إلى لغة الآلة بواسطة المترجم Compiler، ومن ثم يقوم بحفظ البرنامج في الصورة الجديدة حتى يتم تنفيذه بعد ذلك لإخراج النتائج إلى الوسط الخارجي، ليقوم المستخدم بالاستفادة منها بالشكل الذي يريده وذلك عند عدم وجود أخطاء في البرنامج. أما في حالة وجود أخطاء في البرنامج فإنه يجب تصحيح هذه الأخطاء أولاً ثم تعاد الترجمة مرة ثانية وهكذا حتى نحصل على برنامج بدون أخطاء ثم بعد ذلك يتم تنفيذ البرنامج.

الخوارزميات (Algorithms)

لقد استخدمت كلمة الخوارزمية، في القرن الماضي، وبشكل واسع، في أوروبا وأمريكا، وكانت تعني، الوصف الدقيق لتنفيذ مهمة من المهمات، أو حل مسألة من المسائل. وقد اشتق الغربيون هذه الكلمة من اسم عالم الرياضيات المسلم المعروف، محمد بن موسى الخوارزمي.

وتستخدم كلمة الخوارزمية، على نطاق واسع، في علوم الرياضيات والحاسب، الآن حيث تعرف بأنها:
مجموعة الخطوات (التعليمات) المرتبة، لتنفيذ عملية حسابية، أو منطقية، أو غيرها بشكل تتابُعي متسلسل ومنظم.

إن أي خوارزمية تتكون من خطوات مرتبة، بعضها إثر بعض، وكل خطوة تعتبر بنفسها وحدة من وحدات البناء الكامل للخوارزمية، ويختلف حجم هذه الخطوات باختلاف الخوارزميات، واختلاف الأشخاص، الذين يقومون بتنفيذ تلك الخطوات. والمثال التالي يوضح معنى الخوارزمية:
مثال:

إذا أردنا أن نوجد متوسط درجات الحرارة: T_3, T_2, T_1 مثلاً فإن خطوات الحل المنطقية يمكن ترتيبها في الخوارزمية التالية:

الخطوة الأولى: اقرأ قيم درجات الحرارة: T_3, T_2, T_1
الخطوة الثانية: احسب متوسط درجات الحرارة، AV، من المعادلة:

$$AV = (T_1 + T_2 + T_3) / 3$$

الخطوة الثالثة: اطبع النتيجة

مثال آخر:

أراد شخص أن يحسب الزكاة، Z، عن أمواله النقدية، CM، والتي بلغت النصاب الشرعي ، بعد مرور حول قمري عليها، وهي في حوزته، فكيف يفعل ؟

الحل: من المعروف أن قيمة الزكاة تحسب بنسبة 2.5%، من قيمة المال البالغ النصاب ، ولذا فإن خطوات الحل يمكن ترتيبها عل النحو التالي:

الخطوة الأولى: اقرأ قيمة ما بحوزته من مال نقدي بالغ للنصاب ، CM

الخطوة الثانية: احسب قيمة الزكاة المستحقة Z ، من المعادلة Z = .025 CM

الخطوة الثالثة: اطبع النتيجة Z

خرائط التدفق Flow charts

تستخدم خرائط التدفق في بيان خطوات حل المسألة وكيفية ارتباطها ببعض، باستخدام رموز اصطلاحية لتوضيح خطوات الحل، وهذه الرموز مبينة بشكل رقم (2-1)

أهمية استخدام خرائط التدفق:

من أهم فوائد استخدام خرائط التدفق قبل كتابة أي برنامج، الأمور الآتية:

١. تعطي صورة متكاملة للخطوات المطلوبة لحل المسائل في ذهن المبرمج، بحيث تمكنه من الإحاطة الكاملة بكل أجزاء المسألة من بدايتها وحتى نهايتها.

٢. تساعد المبرمج على تشخيص الأخطاء التي تقع عادة في البرامج، وبخاصة الأخطاء المنطقية منها، والتي يعتمد اكتشافها على وضع التسلسل المنطقي، لخطوات حل المسألة لدى المبرمج.

٣. تيسر للمبرمج أمر إدخال أي تعديلات، في أي جزء من أجزاء المسألة، بسرعة، ودون الحاجة لإعادة دراسة المسألة، برمتها من جديد.

٤. في المسائل التي تكثر فيها الاحتمالات والتفرعات، يصبح أمر متابعة دقائق التسلسل، أمراً شاقاً على المبرمج، إذا لم يستعن بمخطط تظهر فيه خطوات الحل الرئيسة بشكل واضح.

الشكل الإصطلاحي (الرمز)	معنى الرمز	
	بداية أو نهاية البرنامج (STRRT / STOP)	(1)
	إدخال أو إخراج (INPUT / OUTPUT)	(2)
	عمليات حسابية وتخزين (CALCULATION AND STORE)	(3)
	تقرير (DECISON)	(4)
	تكرار أو دوران (LOOPING)	(5)
	استدعاء برنامج فرعي (CALL SUBROUTINE)	(6)
	اتجاه سير البرنامج (FLOW LINE)	(7)
	نقطة توصيل وربط (CONNECTOR)	(8)
	تعليق وإيضاح (COMMENT)	(9)

شكل(2-1) الرموز الاصطلاحية لخرائط التدفق

٥. تعتبر رسوم خرائط التدفق المستعملة في تصميم حلول بعض المسائل، مرجعاً، في حل مسائل أخرى مشابهة، ومفتاحاً لحل مسائل جديدة لها علاقة مع المسائل القديمة المحلولة، فتُشبَّه رسوم خرائط التدفق، والحالة هذه، بالرسوم التي يضعها المهندس المعماري عند تصميمه بيتاً أو عمارة، أو مسجداً.... الخ.

أنواع خرائط التدفق

بشكل عام، يمكن القول بأن هناك نوعين، رئيسين من خرائط العمليات وهما:

أ) خرائط سير النظم **System Flowcharts**

يستخدم هذا النوع من الخرائط عند تصميم الأجهزة الهندسية، في المصانع وغيرها، والتي تستعمل أنظمة تحكم ذاتية، مثل العوامة في خزانات المياه، وإشارات المرور الضوئية، وأجهزة ضبط الضغط ودرجات الحرارة في أبراج تقطير البترول، فتعتبر خرائط التدفق هنا، بمثابة المخطط الكامل الذي يبين ترتيب، وعلاقة، ووظيفة، كل مرحلة بما قبلها، وبما بعدها، داخل إطار النظام المتكامل، ويمكن تلخيص الدور الذي تقدمه هذه الخرائط بما يأتي:

١ - تبين موقع كل خطوة من الخطوات الأخرى المكوِّنة للنظام، بحيث يسهل اكتشاف أي خلل يحدث في النظام كله بمجرد النظر، مما ييسر عمليات صيانة الأجهزة، و بأقل التكاليف.
٢ - تسهل إجراء التعديلات التي قد تطرأ مستقبلاً على برنامج النظام في أي موقع منه.
٣ - بيان التفصيلات عن المعطيات المطلوب إدخالها إلى النظام.
٤ - بيان التفصيلات عن أنواع النتائج المتوقعة أو المطلوبة من البرنامج المعد للنظام.
٥ - بيان طرق ربط النظام، ببقية الأنظمة الموجودة في المؤسسة المعنية.

ب) خرائط سير البرامج **Programs Flowchart**

ويستعمل هذا النوع من الخرائط، لبيان الخطوات الرئيسة، التي توضع لحل مسألة ما، وذلك بشكل رسوم اصطلاحية، تبين العلاقات المنطقية، بين سائر خطوات الحل، وموقع ووظيفة كل منها في إطار الحل الشامل للمسألة.

هذا، ويمكن تصنيف خرائط سير البرامج هذه إلى أربعة أنواع رئيسة هي:

١ - خرائط التتابع البسيط **Simple Sequential Flowcharts**

٢ - الخرائط ذات الفروع **Branched Flowcharts**

٣ - خرائط الدوران الواحد **Simple – Loop Flowcharts**

٤ - خرائط الدورانات المتعددة **Multi – Loop Flowcharts**

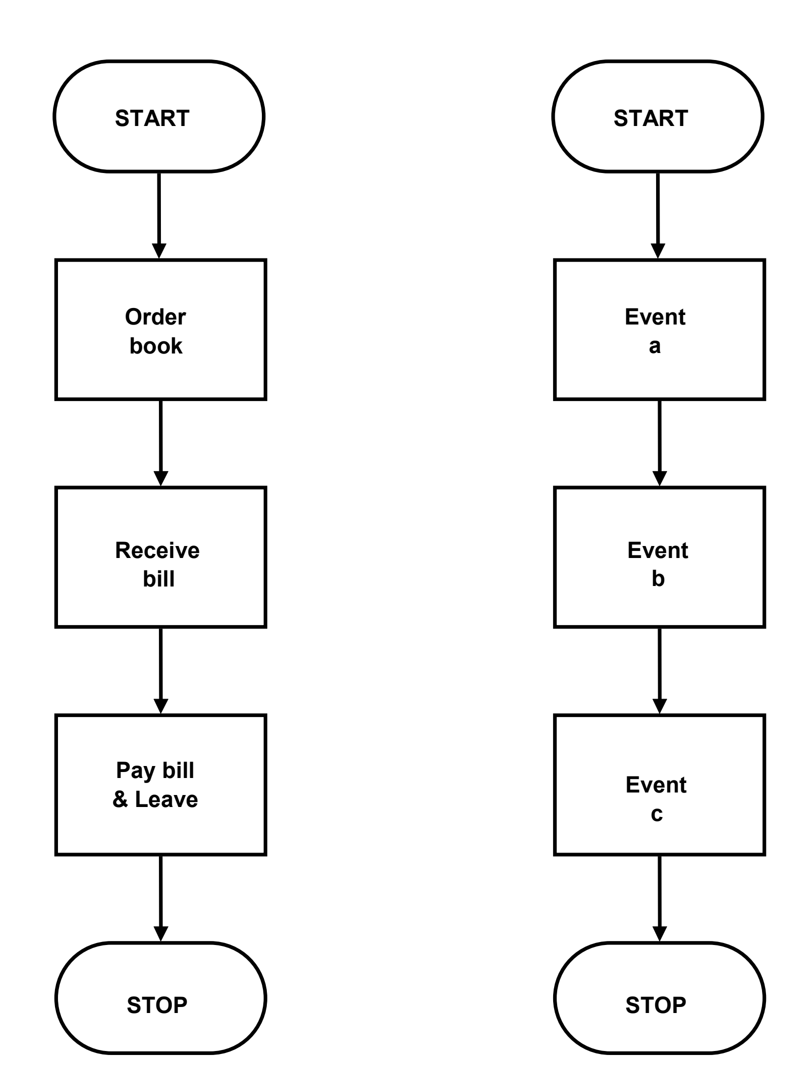
ويمكن للبرنامج الواحد أن يشمل أكثر من نوع واحد من هذه الأنواع، ونتناول فيما يأتي شرح هذه الأنواع بالتفصيل.

خرائط التتابع البسيط

ويتم ترتيب خطوات الحل لهذا النوع من الخرائط، بشكل سلسلة مستقيمة، من بداية البرنامج حتى نهايته، بحيث تنعدم فيها أية تفرعات على الطريق، كما تخلو من أي دورانات مما هو موجود في الأنواع الأخرى من الخرائط. ويكون الشكل العام لهذا النوع كما هو مبين في الشكل (3-1) ، وفيها يتم تنفيذ الحدث a ثم يليه تنفيذ الحدث b وبعده التوقف.
وكلمة الحدث a، الواردة في شكل (3-1) تعني الحدث أو العملية المطلوب تنفيذها.

مثال(1):
ارسم خريطة سير البرنامج التي تمثل عملية شراء كتاب من مركز بيع الكتب.
الحل:
خريطة سير البرنامج في الشكل(4-1) يمكن أن تمثلها الخطوات الآتية:

1- اطلب الكتاب
2- استلم الفاتورة
3- ادفع الفاتورة وغادر



شكل (1-3) خرائط التتابع البسيط

شكل (1-4)

مثال ٢

ارسم خريطة سير البرنامج (flow chart) لإيجاد مساحة ومحيط دائرة نصف قطرها معلوم (R)

الحل:

مساحة الدائرة = πR^2

محيط الدائرة = $2\pi R$

حيث π = النسبة التقريبية وقيمتها العددية ثابتة وتساوي ٣,١٤

بينما R متغير يمثل نصف قطر الدائرة

وحل هذه المسألة كما يأتي :

١- اقرأ قيمة R

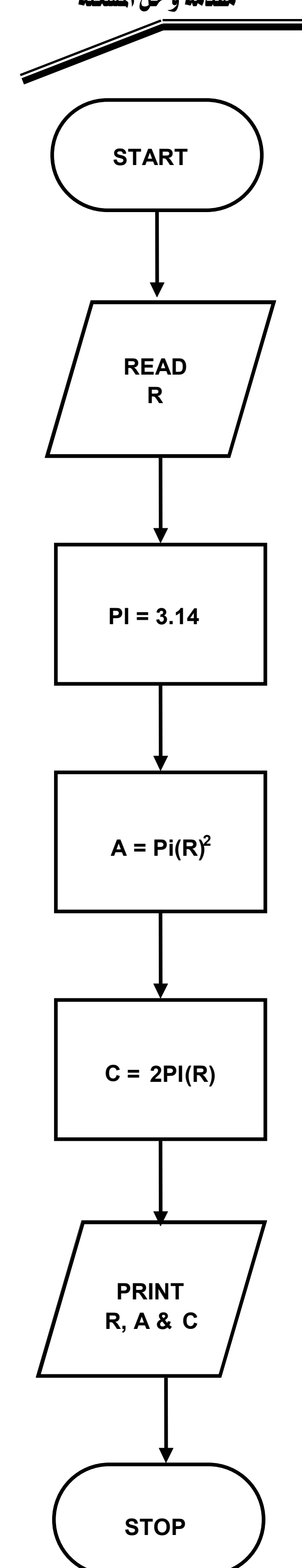
٢- ضع قيمة π = pi= ٣,١٤

٣- احسب مساحة الدائرة A من المعادلة $A = \pi R^2$

٤- احسب مساحة المحيط C من المعادلة $C = 2\pi R$

٥- اطبع قيم كل من : A, R, C

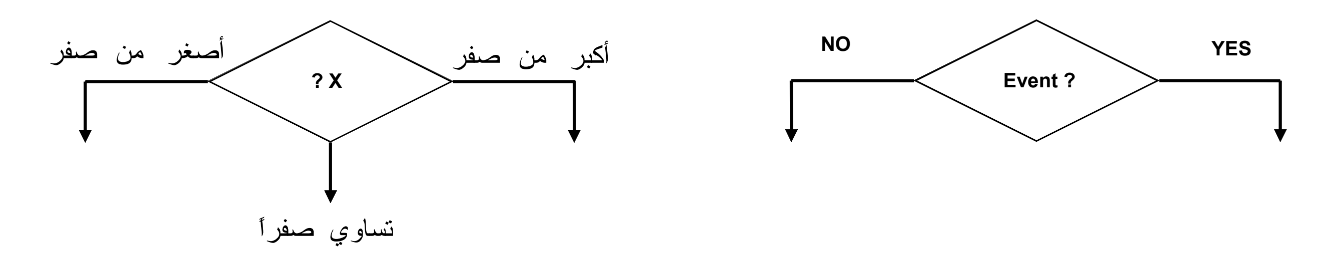
خريطة سير البرنامج التي توضح حل هذه المسألة مبينة في شكل (1-5)



شكل(1-5)

الخرائط ذات الفروع

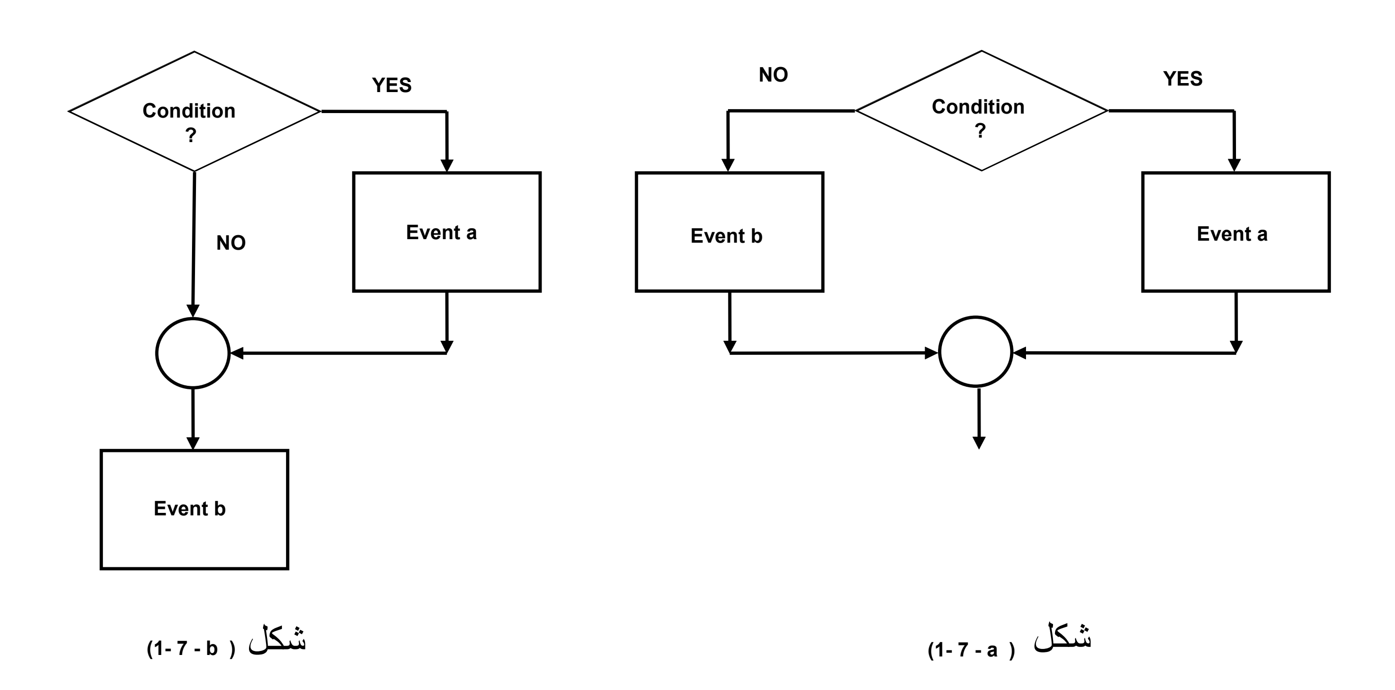
إن أي تفرع يحدث في البرنامج، إنما يكون بسبب الحاجة لاتخاذ قرار، أو مفاضلة بين اختيارين أو أكثر، فيسير كل اختيار في طريق مستقل (تفرع) عن الآخر. وهناك لونان من القرار يمكن للمبرمج استعمال أحدهما حسب الحالة التي يدرسها، والشكل (1-6) يبين هذين المسارين من القرار.



شكل (1-6-a) قرار ذو فرعين

شكل (1-6-b) قرار ذو ثلاثة أفرع

وبشكل عام فإن خرائط التفرع يمكن أن تأخذ إحدى الصورتين الآتيتين كما هو موضح بشكل (1-7). في شكل (1-7-a) يمكن ملاحظة أنه اذا كان جواب الشرط: نعم فإن الحدث التالي في التنفيذ يكون الحدث (a). أما اذا كان الجواب : لا ، فإن الحدث التالي يكون الحدث (b). أما في الشكل (1-7-b) فإننا نلاحظ أنه اذا كان جواب الشرط : نعم، فإن الحدث التالي في التنفيذ يكون الحدث (a) ثم يتبعه الحدث (b)،. أما اذا كان جواب الشرط :لا ، فإن الحدث التالي يكون الحدث (b) مباشرة



شكل (1- 7 - b)

شكل (1- 7 - a)

مثال ٣

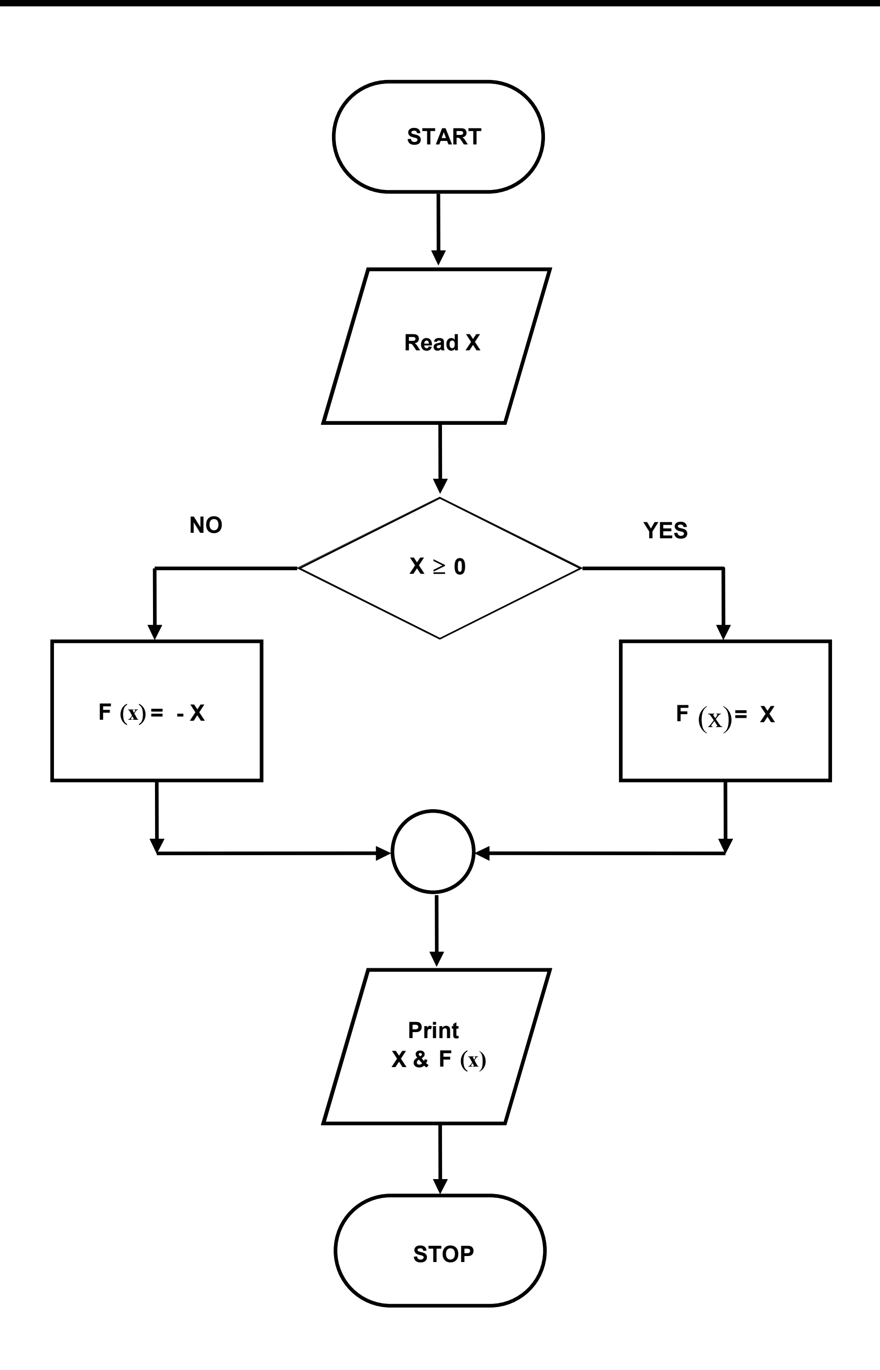
ارسم خريطة سير البرنامج (flow chart) لإيجاد قيمة الدالة F(x) المعرف كما يلي :

$$F(x) = \{ \begin{matrix} x & x \geq 0 \\ -x & x \leq 0 \end{matrix}$$

الحل:

شكل (1-8) يبين خريطة سير البرنامج لحل هذه المسألة كما يلي:

١ - اقرأ قيم المتغير X
٢ - اذا كانت X أكبر من أو تساوي صفرا اذهب إلى خطوة ٣، وإلا فاذهب إلى الخطوة ٤
٣ - احسب قيمة الدالة F(x) من F(x) =X ثم اذهب إلى الخطوة ٥
٤ - احسب قيمة الدالة F(x) من F(x)
٥ - اطبع قيم كل من X , F(x)



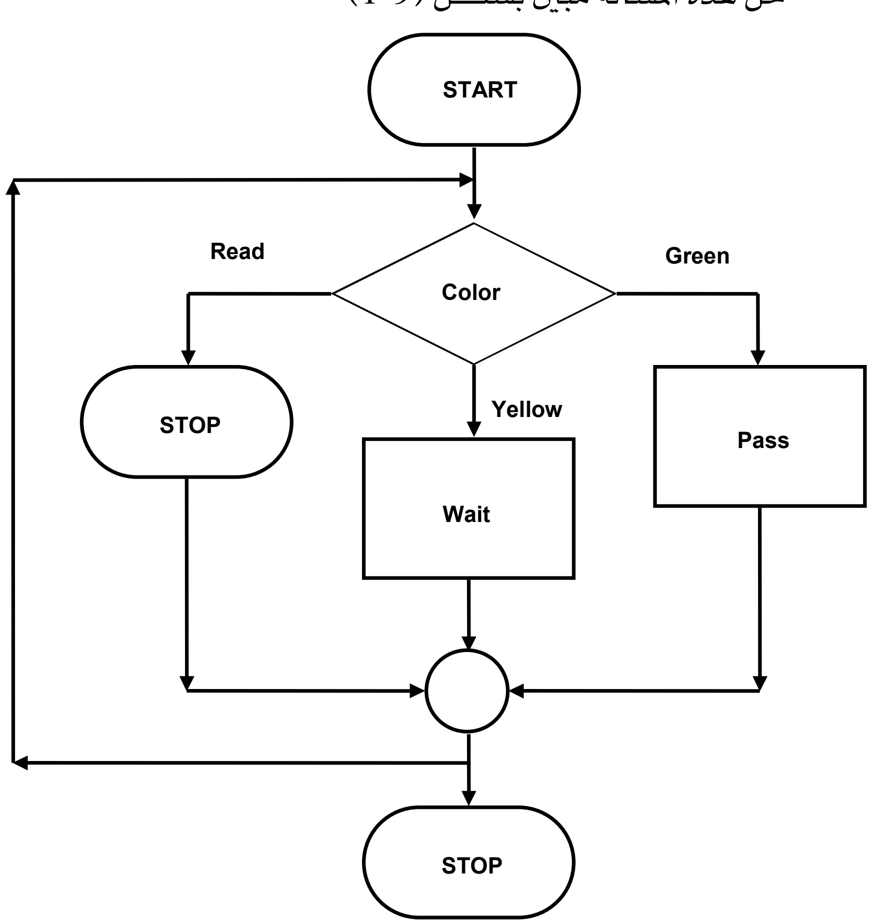
شكل (1-8)

مثال ٤

ارسم خريطة سير البرنامج لإشارات السير الضوئية (اشارات المرور)

الحل:

حل هذه المسألة مبين بشكل (1-9)



شكل (1- 9)

مثال ٥

ارسم خريطة سير البرنامج لحساب قيمة W ، من المعادلات الآتية علماً بأن قيمة المتغير X معلومة

$$W = \begin{cases} X+1 & \text{if } x > 0 \\ \sin(x)+5 & \text{if } x = 0 \\ 2X-1 & \text{if } < 0 \end{cases}$$

الحل :

خطوات الحل مبينة في شكل (1-10) وهي:

١- إذا كانت X أكبر من الصفر اذهب إلى الخطوة ٢

إذا كانت X تساوي صفرا اذهب إلى الخطوة ٣

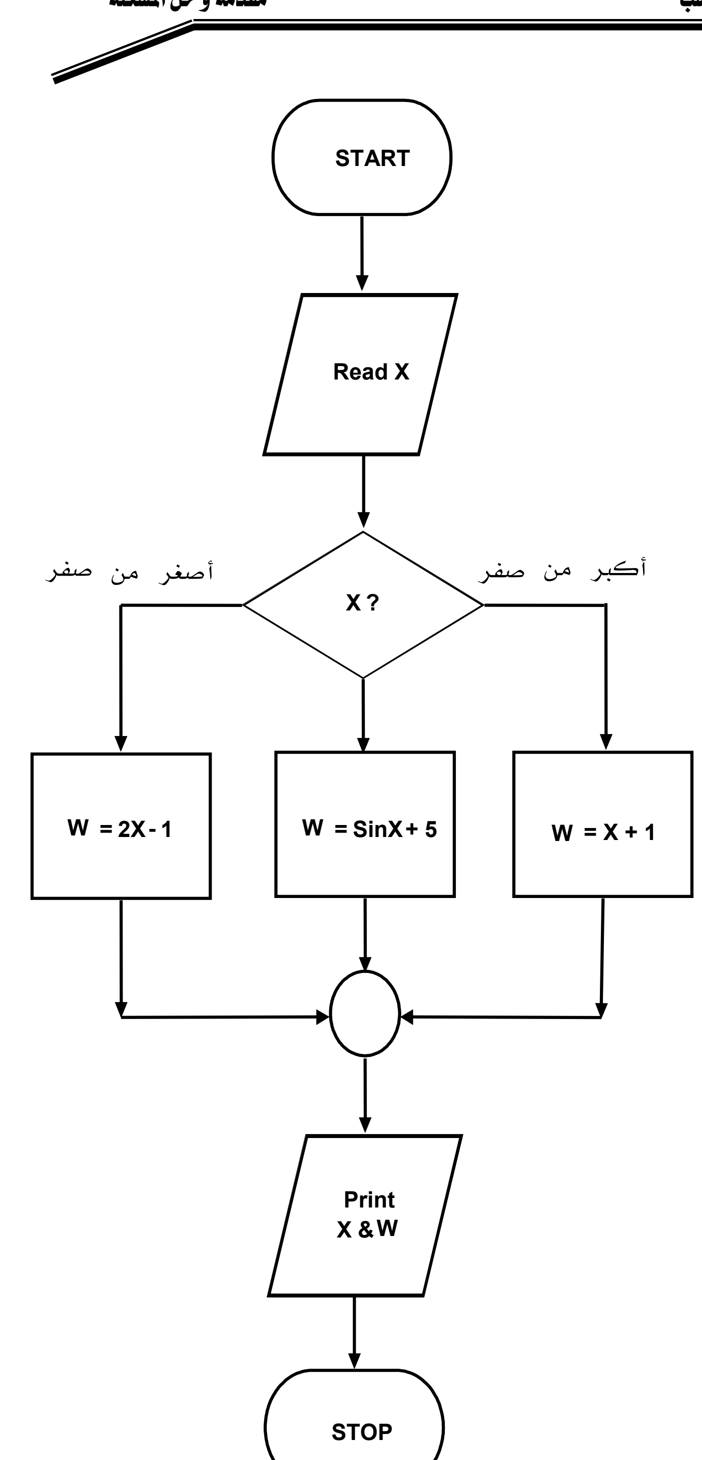
أما إذا كانت X أصغرمن الصفر اذهب إلى الخطوة ٤

٢ - احسب W من المعادلة W= X+1 ثم اذهب إلى الخطوة ٥

٣ - احسب W من المعادلة W= sin (X) +5 ثم اذهب إلى الخطوة ٥

٤ - احسب W من المعادلة W=2X -1

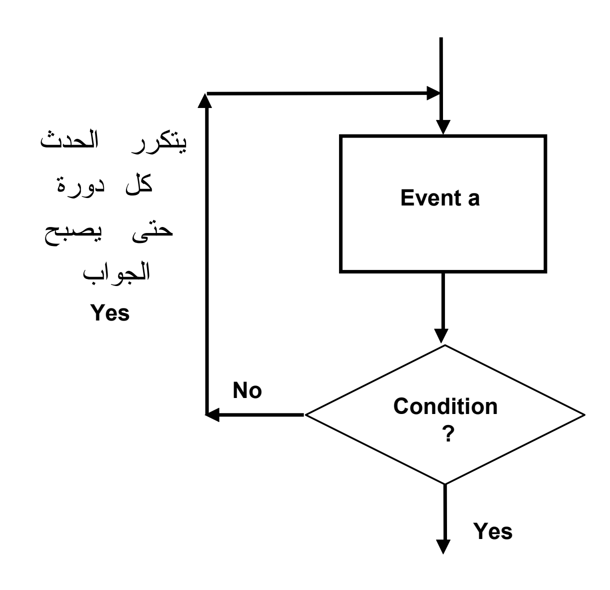
٥ - اطبع قيم كل من X, W

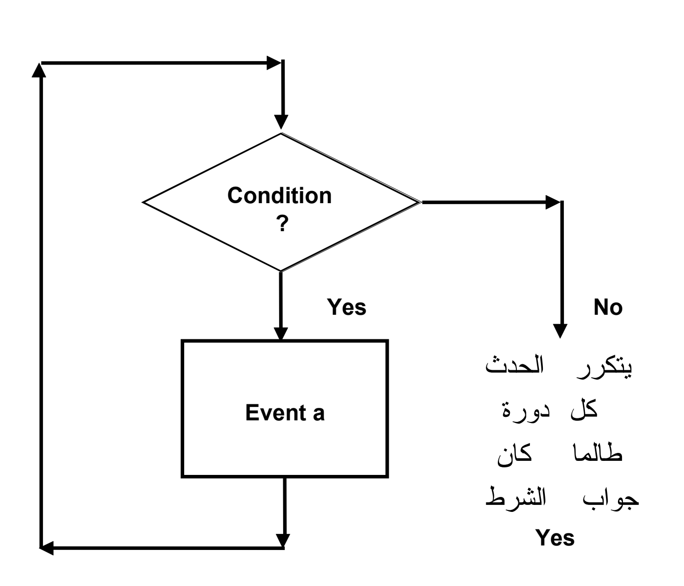


شكل (1-10)

خرائط الدوران الواحد:

وهذه الخرائط نحتاج اليها لإعادة عملية أومجموعة من العمليات في البرنامج عددا محدودا أو غير محدود من المرات، والشكل العام لمثل هذه الخرائط مبين بشكل (1-11). وقد سميت هذه الخرائط بخرائط الدوران الواحد لأنها تستعمل حلقة واحدة، وتسمى أحيانا خرائط الدوران البسيط،





شكل(1-11)

مثال ٦

ارسم خريطة سير البرنامج لإيجاد مساحة مجموعة من الدوائر أنصاف أقطارها معلومة

الحل : خطوات الحل مبينة في شكل (1-12) وهي:

١ - اقرأ نصف قطر الدائرة R
٢ - أوجد مساحة الدائرة A
٣ - اطبع قيم كل من A,R
٤ - هل هناك المزيد من الدوائر ؟
إذا كان نعم عد للخطوة ١
أما إذا كان لا فتوقف

مثال ٧

من واجبات بيت مال المسلمين أن يساعد الشباب على الزواج وذلك بتقديم الدعم المادي المناسب لهم (٣٠٠٠٠ ريال للزيجة الواحدة) على فرض أن السن المناسب للزواج هو عشرون عاما ، اقترح خريطة لسير البرنامج لهذا المشروع.

الحل: خطوات الحل مبينة في شكل (1-13) وهي:

١- خذ شابا على الدور

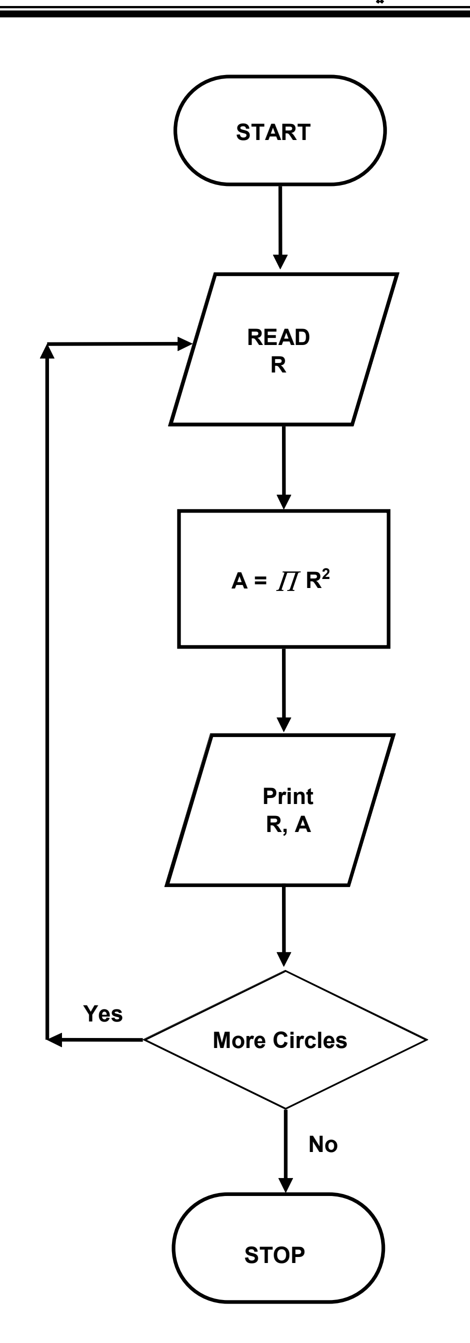
٢- هل مضى من عمره عشرون عاما؟
إن كان نعم اذهب الى الخطوة ٣
أما إن كان لا ، اذهب إلى الخطوة ٤

٣- زوج الشاب المذكور

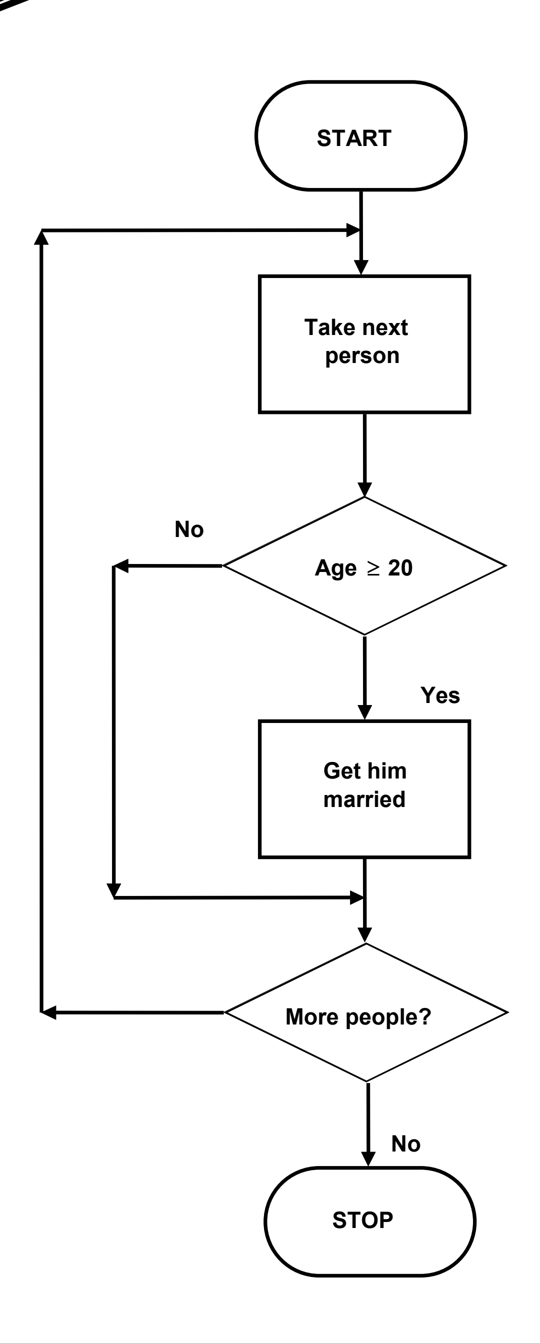
٤- هل هناك مزيد من الشباب؟
إن كان نعم اذهب الى الخطوة ١
أما إن كان لا ، فتوقف

ملحوظة:

ينبغي التنبيه هنا إلى أن عملية الانتقال من خطوة ٢ إلى الخطوة ٤ – عندما تكون الاجابة "لا" – لا تمثل دورانا أو تكرارا لأن عملية الدوران إنما تتم بالانتقال من خطوة متأخرة إلى خطوة متقدمة عدة مرات لإعادتها ، ولذا فإن هناك دورانا بسيطا واحدا في هذا المثال، ويمثله العودة من خطوة ٤ إلى خطوة ١



شكل (1-12)



شكل (1-13)

مثال ٨

ارسم خريطة سير البرنامج لخزان ُيملأ بالماء ذاتيِّاً (أتوماتيكيِّاً)، عندما يصبح ارتفاع مستوى الماء فيه أقل من متر.

الحل : من المعلوم أن عملية ملء الخزان تقوم على فكرة وجود العوامة التي تفتح صنبور التغذية ذاتياً عندما يصل ارتفاع الماء حداً معيناً (متراً واحداً في هذا المثال) وتغلق صنبور التغذية عند وصول مستوى الماء في الخزان إلى الارتفاع المطلوب وبالتالي فإن خطوات الحل المبينة في الشكل (1-14) تكون كما يأتي:

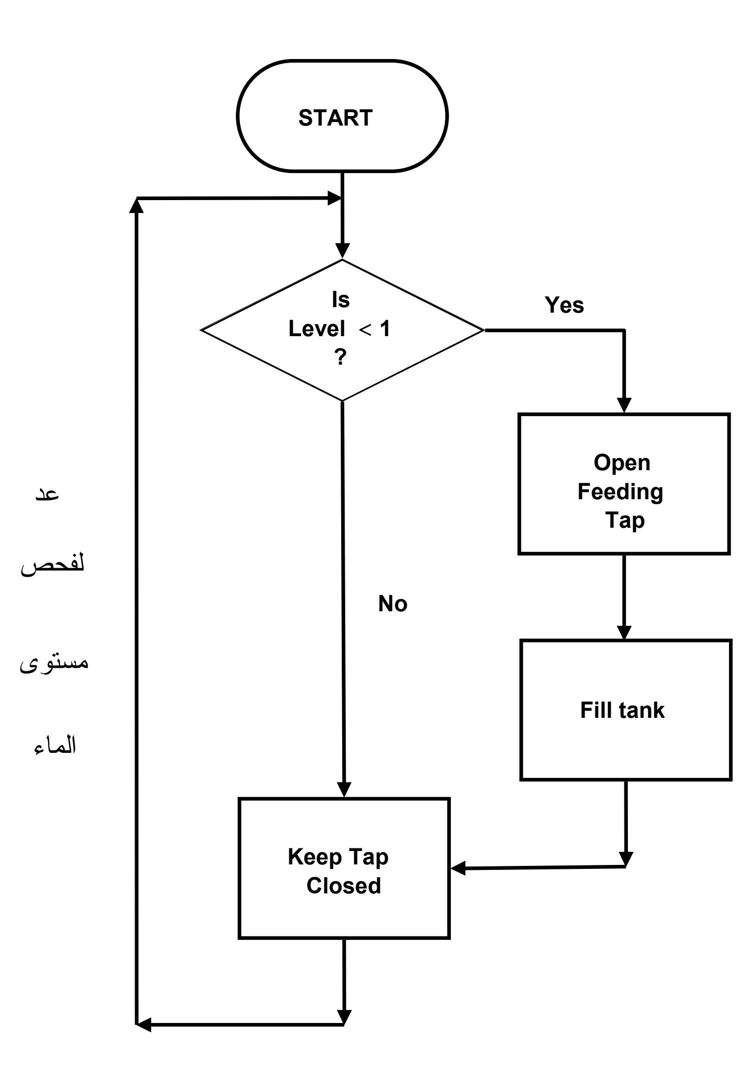
1- هل مستوى الماء أقل من متر ؟ إذا كان الجواب نعم فاذهب إلى الخطوة (2) وإذا كان الجواب لا ، فاذهب إلى الخطوة (4)

2- يفتح صنبور التغذية.

3- يملأ الخزان إلى المستوى المطلوب.

4- أغلق الصنبور (أو حافظ عليه مغلقاً).

5- عد إلى الخطوة (1) لفحص مستوى الماء مرة بعد مرة للحفاظ على الوضع المطلوب وبشكل دائم.



شكل (1-14)

مثال ٩

الشكل (1-15) يمثل خريطة سير البرنامج لمجموعة من العمليات الحسابية. ادرس العمليات بعد تتبع الخريطة.

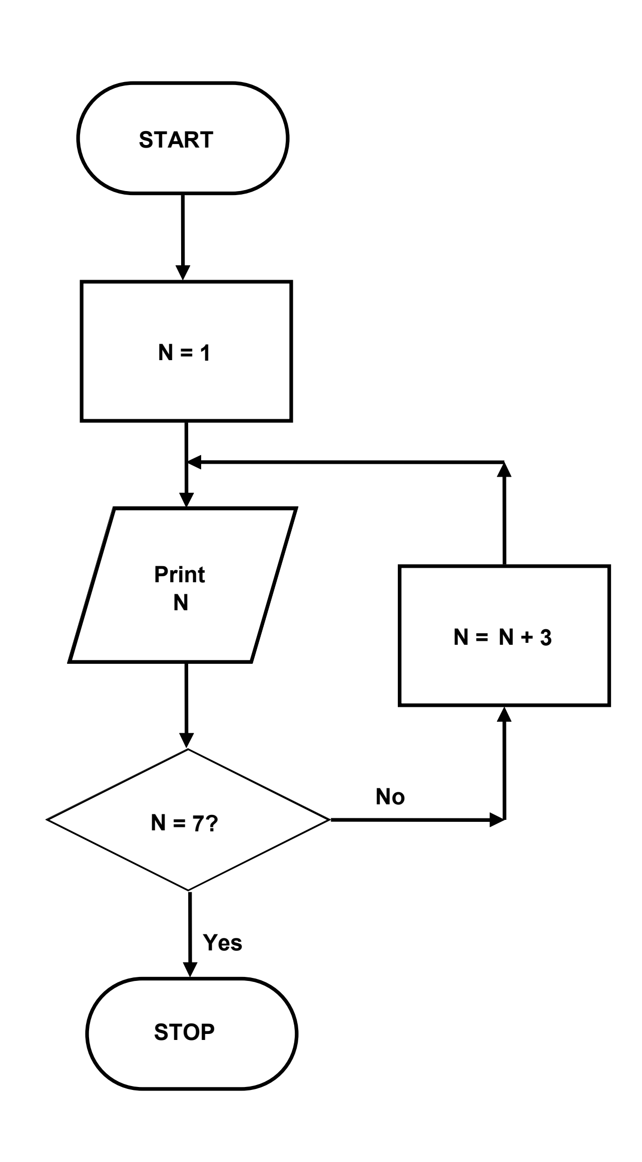
من الشكل نلاحظ أن الحاسب يبدأ بوضع قيمة مبدئية أولى مقدارها 1 في مخزن الذاكرة N كما في الشكل (1-16a)، ثم يقوم بطبع هذه القيمة، من خلال جهاز الإخراج، وعلى الوسط الخارجي ثم يسأل (هل القيمة المخزونة في مخزن N تساوي 7؟) الجواب بالطبع : لا، لأن $7 \neq 1$، لهذا فهو ينفذ الأمر التالي: $N = N + 3$ وهذا الأمر يعني أن الحاسب سيضع في مخزن N ما كان فيه سابقاً مضافاً إليه 3 لتصبح القيمة المخزونة في المخزن N تساوي 4 بدلاً من 1 (انظر الشكل (1-16b)) ثم يعود مرة أخرى بعد أن يطبع N الجديدة على الوسط الخارجي ليسأل هل $7 = 4$ ؟ ويكرر العملية السابقة حتى تصبح القيمة المخزونة في المخزن N تساوي 7 وعندها يتوقف البرنامج بالأمر: توقف، ويكون قد طبع لنا على الوسط الخارجي القيم التي خزنت في مخزن N على التوالي وهي:

1

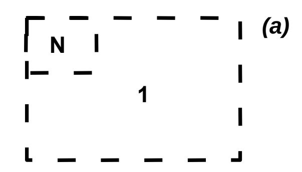
4

7

- ملحوظة: هناك مخزن واحد فقط تحت اسم N في وحدة الذاكرة تخزن فيه قيمة واحدة في الوقت الواحد، ولذا فإن آخر قيمة تبقى في المخزن N في المثال هي 7



شكل (1-15)



N
4
(b)

N
7
(c)

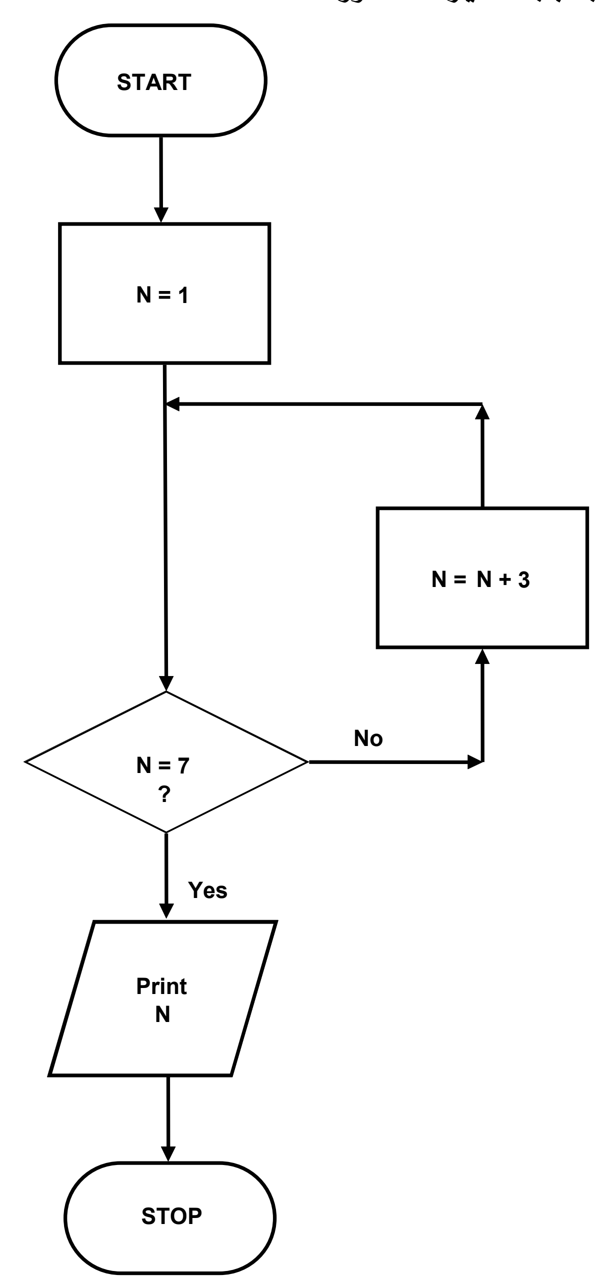
شكل (1-16)

مثال ١٠

لو أحدثنا تغييراً بسيطاً في الشكل (1-15) ليصبح كما هو مبين في شكل (1-17) فما أثر ذلك التغيير؟

نلاحظ من الشكل (1-15) أن التغيير الذي حدث يتلخص في أن خطوة كتابة قيمة المتغير N قد تأخرت عن خطوة التقرير (هل N = 7)؟ وهذا يعني أن كتابة قيمة المتغير N تأتي بعد الانتهاء من الدوران أي بعد أن تصبح قيمة N تساوي 7، ولذا فإن نتائج الإخراج تكون قيمة واحدة فقط وهي: 7

في حين أن نتائج الإخراج في المثال السابق كانت تطبع في كل دوران، مما جعل النتائج في المثالين مختلفة بسبب التغير المذكور.



شكل (1-17)

مثال ١١

ارسم خريطة سير البرنامج لإيجاد مجموع m من الأعداد الحقيقية

$(X_1, X_2, ..., X_m)$

الحل: النتيجة المطلوبة هي مجموعة الأعداد T حيث إن $T = \sum_{i=1}^{m} Xi$

خطوات الحل يمكن أن تسير على النحو التالي:

$$T_0 = 0$$
$$T_1 = T_0 + X_1 = 0 + X_1 = X_1$$
$$T_2 = T_1 + X_2 = X_1 + X_2$$
$$T_m = T_{m-1} + X_m = X_1 + X_2 + ... + X_{m-1} + X_m$$

ونموذج الحل هذا يمكن أن يختصر بنموذج مكافئ هو:

$$T_i = T_{i-1} + X_i \qquad (1)$$

حيث $T_0 = 0$ وتتغير i من 1 إلى m

ويمكننا اختصار عدد المتغيرات T_1 .... T_m بمتغير واحد هو T، الذي تكون قيمته في كل دوران مساوية لقيمته السابقة مضافاً إليها قيمة X المراد إضافتها إليه. وذلك بإعادة كتابة المعادلة (1) على النحو:

$$T_i = T + X_i \qquad i = 1, m \qquad (2)$$

على أن تكون القيمة الأولى للمجموع T تساوي صفراً.

وترتيب النموذج في المعادلة (2) من شأنه أن يوفر أكبر عدد ممكن من المخازن الشاغرة في الذاكرة، لاستخدامها في أغراض أخرى، وكذلك فإن صيغة المعادلة (2) أسهل من صيغة المعادلة (1) وبالتالي فإنها تساعد على تسهيل عملية البرمجة.

أما خريطة سير البرنامج فمبيَّنة في شكل (18-1) (افرض أن قيمة m تساوي 100 هنا)

خرائط الدورانات المتعددة

في هذه الحالة تكون الدورانات داخل بعضها البعض بحيث لا تتقاطع فإذا كان لدينا دورانان من هذا النوع (انظر شكل (1-19)) فيسمى الدوران رقم (1) دوراناً داخلياً Inner Loop بينما الدوران رقم (2) دوراناً خارجياً Outer Loop، ويتم التنسيق بين عمل مثل هذين الدورانين، بحيث تكون أولوية التنفيذ للدوران الداخلي.

وقد سميت هذه الخرائط بخرائط الدورانات المتعددة لأنها تستعمل أكثر من حلقة دوران واحدة، وقد تسمى أحياناً بخرائط الدورانات المتداخلة أو المتراكبة أو الضمنية nested، وكل هذه التسمية تؤدي إلى معنى واحد.

شكل (1-18)

شكل(1-19)

مثال ١٢

يرغب تاجر في تقطيع مجموعة من قطع القماش طول كلٍ منها يزيد عن 5 أمتار، إلى قطع صغيرة، طول الواحدة منها يساوي 5 أمتار، ارسم خريطة سير البرنامج لهذا المشروع.

خطوات الحل المبينة في شكل (20-1) هي:

1- خذ قطعة

2- اقطع منها قطعة طولها 5 متر

3- هل المتبقي يزيد عن 5 متر؟

إذا كان الجواب نعم، فاذهب إلى الخطوة (2)

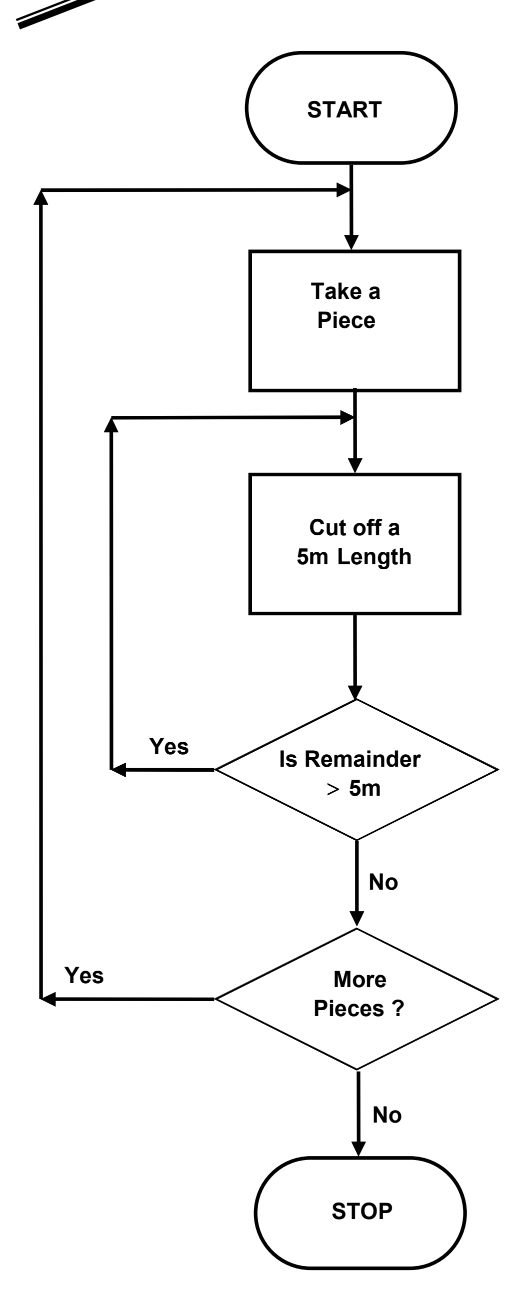
إذا كان الجواب لا، فاذهب إلى الخطوة (4)

4- هل هناك مزيد من القطع المراد تقطيعها؟

إن كان الجواب نعم، فاذهب للخطوة (1)

وإن كان لا، فتوقف

- **ملحوظة:** يلاحظ من الشكل (20-1) أن الدوران الداخلي يتضمن تقطيع القطعة الواحدة إلى قطع متعددة، طول كل منها 5 متر، بينما يمثل الدوران الداخلي تناول قطعة واحدة جديدة لتنفذ عليها إجراءات الدوران الداخلي.

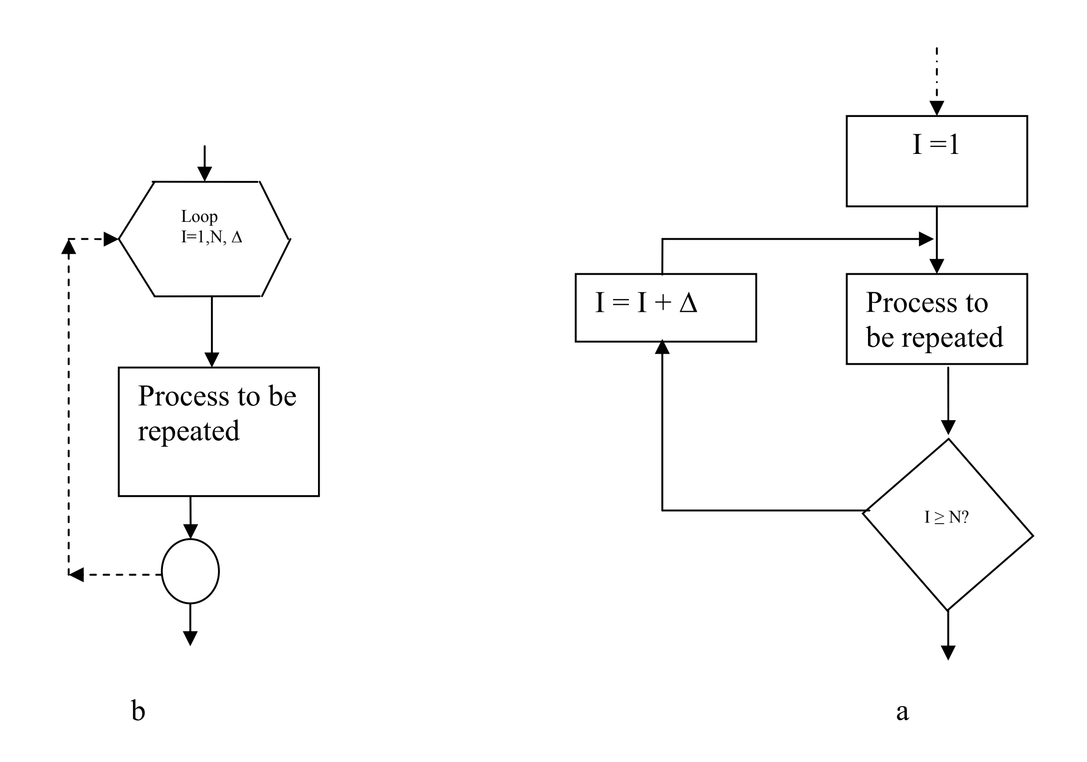


شكل (20)

صيغة الدوران باستعمال الشكل الاصطلاحي

لقد عرفنا في الفقرتين السابقتين مفهوم الدوران البسيط والدورانات الضمنية، ويمكننا الآن استخدام الشكل الاصطلاحي للدوران -الوارد ضمن الرموز الاصطلاحية لخرائط سير البرنامج - على النحو التالي:

نلاحظ في الشكل (1-21) أننا نحتاج إلى العناصر الآتية:



شكل (1-21)

العدّاد(I)

القيمة الأولية للعدّاد I (هنا i = 1)

القيمة النهائية للعدّاد I (هنا N)

قيمة الزيادة في العدّاد عند نهاية كل دورة (Δ)

نلاحظ من الشكل (1-21-a) أن إجراءات الدوران كانت تتم طبقاً للخطوات الآتية والمفصلة من قبل المبرمج:

1- أعط I قيمة أولية.

2- أتم الإجراءات المطلوب إعادتها.

3- اتخاذ قرار: إذا كانت قيمة العداد I وصلت إلى القيمة النهائية N، فاخرج إلى الخطوة التالية في البرنامج وإلا فاذهب إلى الخطوة (4)

4- زد العداد بمقدار الزيادة Δ

5- عد إلى (2)

يمكننا استبدال الخطوات المفصلة (5,4,3,1) في الشكل (1-21-a) بخطوة مجملة واحدة مبينة في الشكل الاصطلاحي للدوران شكل (1-21-b)، حيث تنفذ هذه الخطوات بصورة أوتوماتيكية من قبل الحاسب. وهذا من شأنه تسهيل عملية البرمجة، واختصار عدد العمليات في البرنامج وتجنب بعض الأخطاء.

- ملحوظة: تعتبر قيمة Δ تساوي 1 دائماً إذا لم تُعطَ قيمة أخرى بخلاف ذلك، وفي حالة عدم ذكر قيمة Δ يصبح الشكل الاصطلاحي الوارد في شكل (1-21-b) كما هو موضح بشكل (1-22) وتكون قيمة الزيادة Δ تساوي 1، بصورة أتوماتيكية.

Loop
I = 1, n

شكل (1-22)

مثال ١٣

أعد حل مثال (6) لإيجاد مساحة من الدوائر باستخدام الشكل الاصطلاحي للدوران.

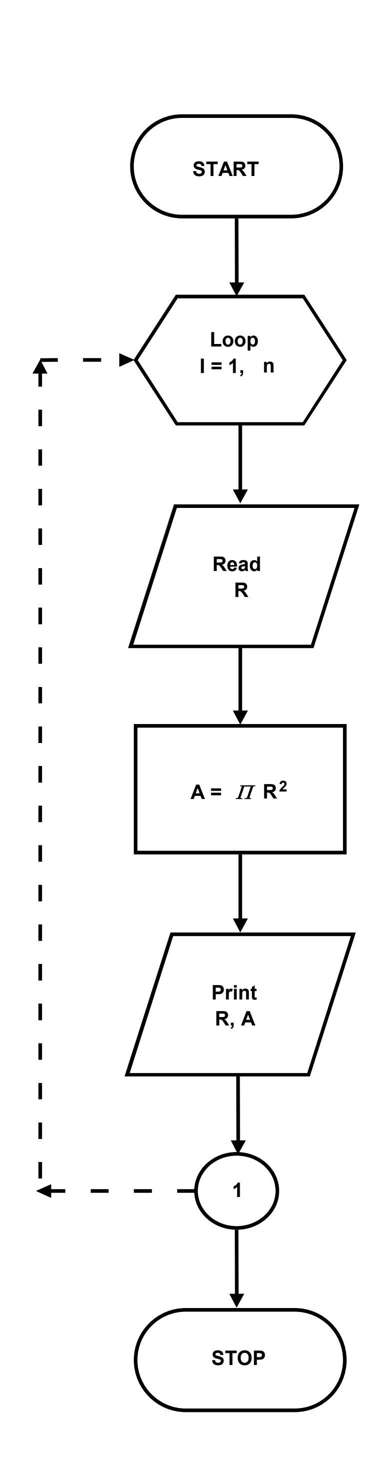
خطوات الحل كما مبينة في الشكل (1-23)

مثال ١٤

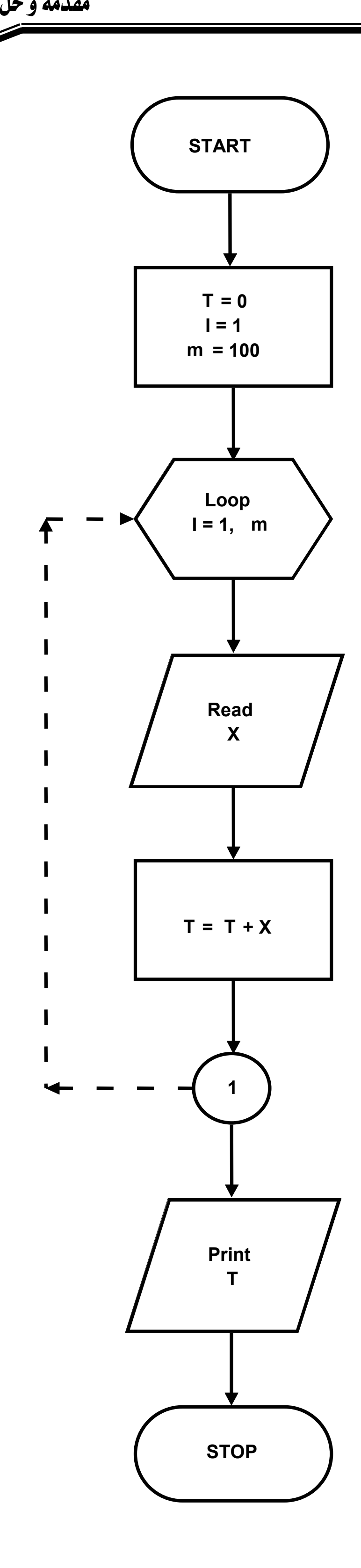
أعد حل مثال (11) باستخدام الشكل الاصطلاحي للدوران. بحيث

$m = 100$

خطوات الحل كما هي مبينة في الشكل (1-24)



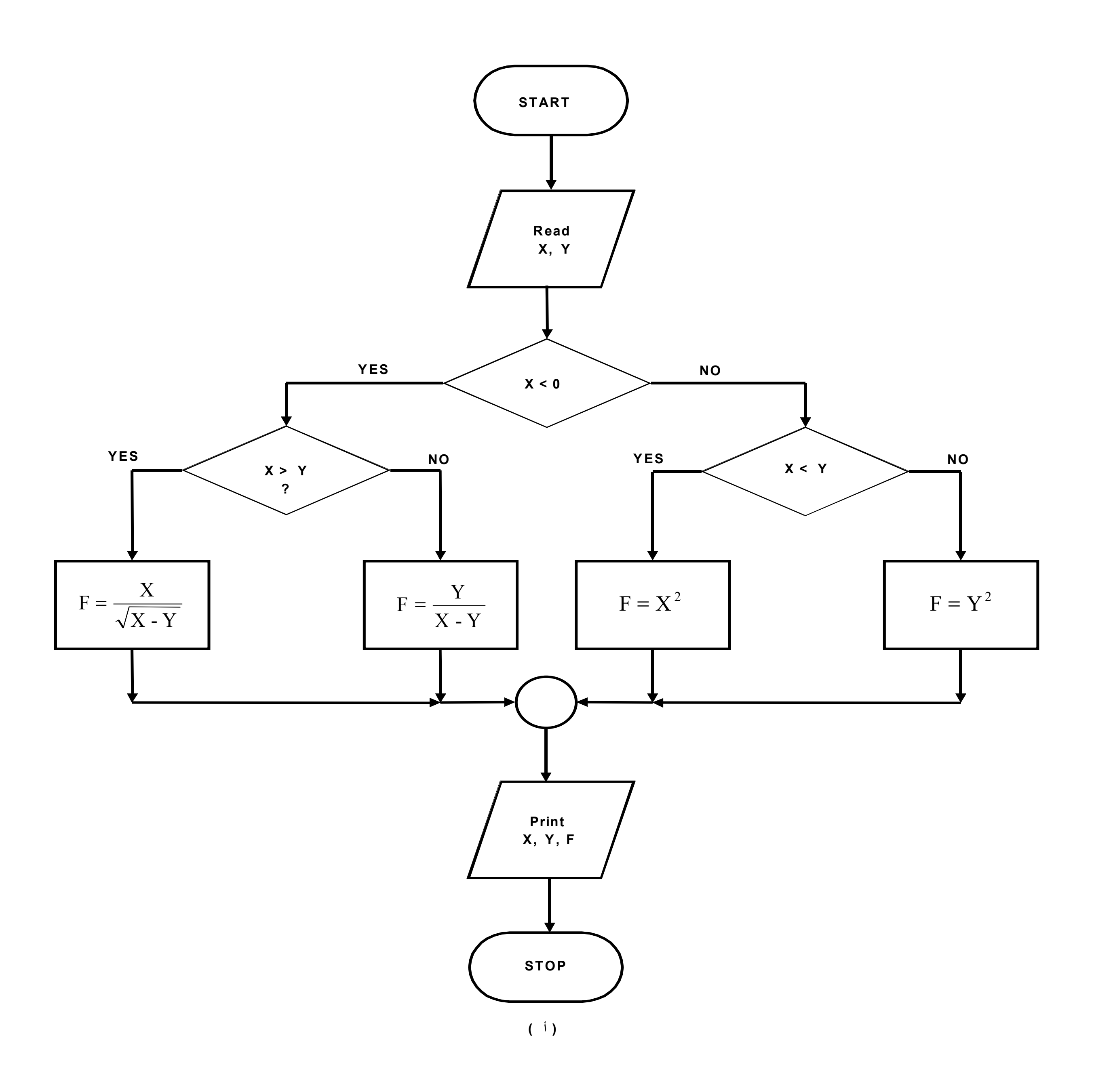
شكل (1-23)



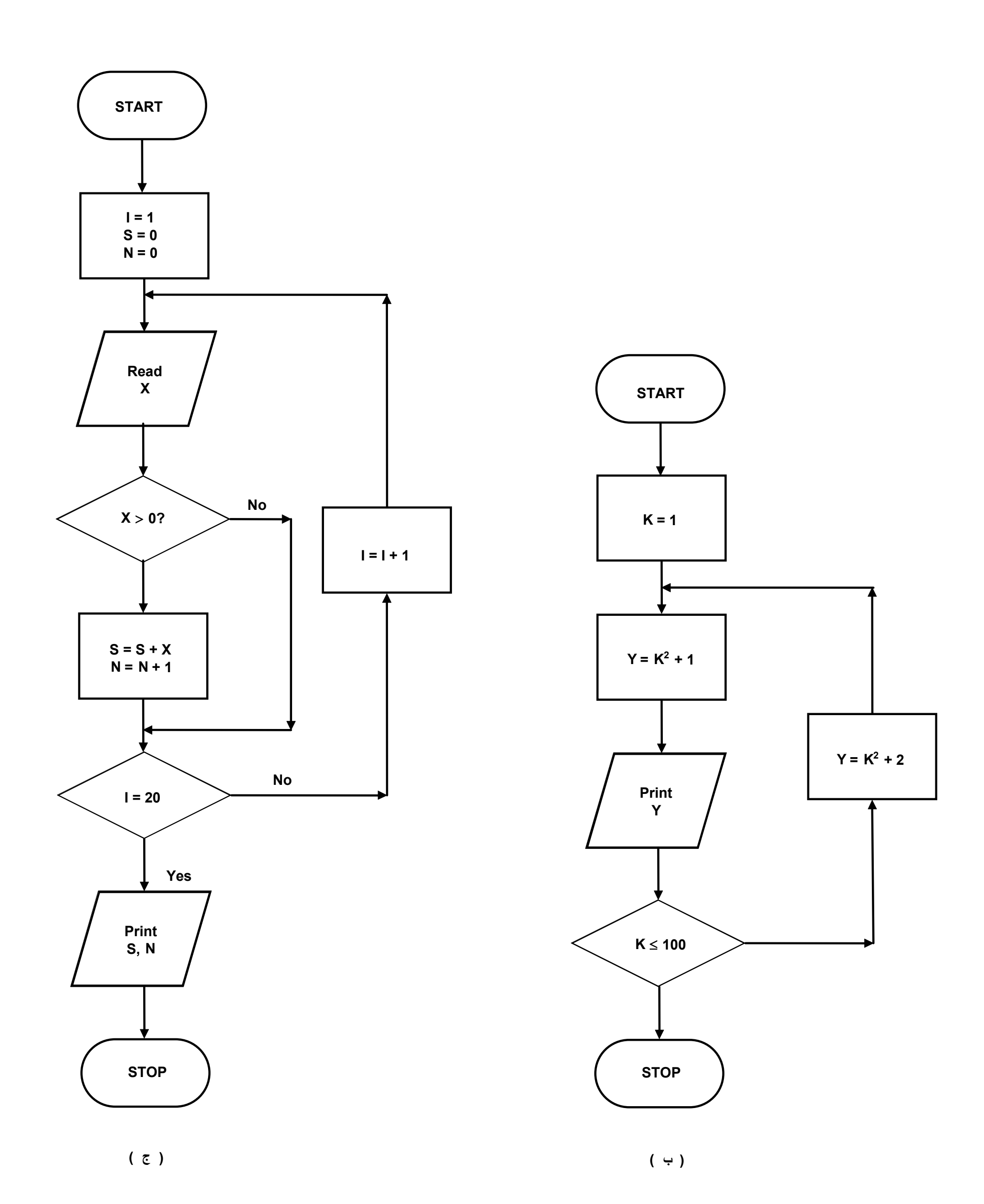
شكل (1-24)

تدريبات

(١) ادرس المخططات (أ،ب،ج) مبيناً أهداف كل مخطط والنواتج النهائية التي سيطبعها الحاسب عند تنفيذ التعليمات المبينة إزاء كل مخطط ، x=3 , Y=5 .



(أ)


START
I = 1
S = 0
N = 0
Read
X
X > 0?
No
S = S + X
N = N + 1
I = I + 1
I = 20
No
Yes
Print
S, N
STOP
START
K = 1
Y = K2 + 1
Print
Y
K ≤ 100
Y = K2 + 2
STOP

(ج)   (ب)

٢- ارسم خريطة سير البرنامج التي تمثل كلا من الخوارزميات التالية:

(أ)

1- ضع قيمة SUM صفراً، وقيمة N تساوي 1

2- اجمع N إلى SUM

3- إذا كانت N<6 فأضف 1 إلى قيمة N الحالية، ثم اذهب إلى الخطوة 1

4- اطبع قيمة SUM

(ب)

1- اقرأ قيمة X

2- إذا كانت $X \geq 0$ فاذهب إلى الخطوة (5)

3- احسب قيمة W من المعادلة $W = \sqrt{x^2 + 5x - 4}$ ، ثم اذهب إلى الخطوة (5)

4- احسب قيمة W من المعادلة: $W = -X + 13$

5- اطبع قيمتي X و W

3- ارسم خريطة سير البرنامج لحساب كلٍ من الاقترانات الآتية:

(أ) $f(X) = | X-3 |$

(ب) $SUM = \sum_{i=1}^{n} i$

(ج) $F = n! = n(n-1) \dots (2)(1)$

(د) إيجاد قيمة أكبر عدد في المجموعة S حيث:

$$S = [A, B, C]$$

(هـ) إيجاد قيمة أصغر عدد في المجموعة S نفسها تنازلياً ثم تصاعدياً.

(ز) إيجاد قيمة أكبر عدد في السلسلة الحسابية:

$$a_1, a_2, a_3, \dots, a_{n-1}, a_n$$

(ح) ارسم خريطة سير البرنامج لإيجاد قيمة أصغر عدد في المتسلسلة الحسابية في السؤال السابق.

(ط) ارسم خريطة سير البرنامج بحيث ترتب حدود المجموعة التالية ترتيباً تنازلياً:

$$a_1b_1, a_2b_2, a_3b_3, \dots, a_{n-1}b_{n-1}, a_nb_n$$

(ى) ارسم خريطة سير عمليات لترتيب حدود المجموعة ترتيباً تصاعدياً.

(ل) اكتب أول 200 حد في المتوالية الهندسية التي تبدأ بالعدد 5، بحيث يكون معدل التغير 3.

(م) اكتب أول ثلاثين حداً في المتسلسلة التالية:

1, 3, 5, 7, ….

(ن) أوجد قيمة الاقتران عديد الحدود :poly حيث:

$$POLY = 1+Z+Z^2+ \dots +Z^{10}$$

إذا كانت قيمة المتغير Z معروفة لديك.

يعبأ هذا النموذج عن طريق المدرب

اسم المتدرب : - - - - - - - - **التاريخ** - - - - - -

رقم المتدرب : - - - - - - - - - - - **المحاولة ١ ٢ ٣ ٤**

كل بند أو مفردة يقيم بـ ١٠ نقاط

العلامة : الحد الأدنى : ما يعادل ٨٠ ٪ من مجموع النقاط

الحد الأعلى : ما يعادل ١٠٠٪ من مجموع النقاط

بنود التقييم	النقاط
• تحديد أجزاء المشكلة الرئيسة • تحديد أجزاء المشكلة الفرعية • تقسيم المشكلة إلى أجزاء صغيرة • تحديد احتياجات حل المشكلة • معرفة رموز رسم خرائط التدفق • رسم خرائط التدفق للمشاكل البسيطة • رسم خرائط التدفق للمشاكل المتوسطة • وضع خوارزمية الحل للمشاكل المتوسطة	

ملاحظات: -

توقيع المدرب: -

يعبأ هذا النموذج عن طريق المتدرب

تعليمات

بعد الانتهاء من التدريب على **حل المشكلة** قيِّم نفسك بواسطة إكمال هذا التقييم الذاتي لكل عنصر من العناصر المذكورة، وفي حالة عدم قابلية المهمة للتطبيق ضع العلامة في الخانة الخاصة بذلك.

اسم النشاط التدريبي الذي تم التدرب عليه: **حل المشكلة**

العناصر	مستوى الأداء (هل أتقنت الأداء)			
	غير قابل للتطبيق	لا	جزئيا	كليا
• تحديد أجزاء المشكلة الرئيسة • تحديد أجزاء المشكلة الفرعية • تقسيم المشكلة إلى أجزاء صغيرة • تحديد احتياجات حل المشكلة • معرفة رموز رسم خرائط سير البرنامج • رسم خرائط التدفق للمشاكل البسيطة • رسم خرائط التدفق للمشاكل المتوسطة • وضع خوارزمية الحل للمشاكل المتوسطة والبسيطة				

يجب أن تصل النتيجة لجميع المفردات (البنود) المذكورة إلى درجة الإتقان الكلي أو أنها غير قابلة للتطبيق، وفي حالة وجود مفردة في القائمة "لا" أو جزئيا فيجب إعادة التدرُّب على هذا النشاط مرة أخرى بمساعدة المدرب

المملكـة العربيـة السـعودية
المؤسسة العامة للتعليم الفني والتدريب المهني
الإدارة العامة لتصميم وتطوير المناهج



برمجة الحاسب

مكونات لغة الجافا

مكونات لغة الجافا

٢

الجدارة:

أن يكون المتدرب قادرا على كتابة الشفرة البرمجية code للبرامج البسيطة نسبيا

الأهداف :

عندما تكمل هذه الوحدة يكون لديك القدرة على:

١ - فهم ومعرفة واستخدام المتغيرات والثوابت
٢ - معرفة وكتابة جمل التعليق المختلفة
٣ - كتابة برامج بسيطة بلغة الجافا
٤ - استخدام جمل الإسناد
٥ - فهم واستعمال الأنواع المختلفة للبيانات مثل الأعداد الصحيحة والأعداد العشرية بأحجامها المختلفة بالإضافة إلى النصوص والأحرف والأعداد المنطقية
٦ - معرفة واستخدام العوامل المختلفة وكتابة التعبيرات البرمجية بلغة الجافا
٧ - كتابة التعبيرات الحسابية والمنطقية المختلفة
٨ - معرفة واستخدام أولويات تنفيذ العمليات المختلفة
٩ - تحويل الأعداد من نوع إلى نوع آخر
١٠ -استخدام الأقواس

مستوي الأداء المطلوب:

أن يصل المتدرب إلى إتقان هذه الجدارة بنسبة 100 ٪

الوقت المتوقع للتدريب : ٢٤ ساعة

الوسائل المساعدة:

- حاسب إلى
- قلم
- دفتر

متطلبات الجدارة:

اجتياز جميع الحقائب السابقة

مكونات لغة الجافا

Components of Java programming languages

هذه الوحدة تبحث في أبجديات مكونات لغة الجافا (java) والتي تتكون من المتغيرات (Variables) والثوابت (Constants) والكلمات المحجوزة (Reserved Words) وغيرها والتي سوف نعرضها في هذه الوحدة . وسوف نتعرض بالشرح أيضا لجميع العمليات الحسابية(arithmetic operation) والمنطقية (logic) وعمليات الإسناد(assignment) والعمليات العلاقية (Relational operation) والنصية (String) وغيرها من العمليات.

أولا : تمثيل البيانات الأولية

في هذا الجزء من هذه الوحدة سوف نتعرف على بعض العناصر الأساسية والتي تستخدم في بناء برامج الجافا مثل المتغيرات والثوابت وغيرها. وسوف نقوم بشرح هذه المكونات من خلال أمثلة مكتوبة بلغة الجافا. وقبل الحديث عن مكونات لغة الجافا يجب أن نعلم أن البرامج المكتوبة بلغة الجافا تنقسم إلى نوعين : -

النوع الأول فيها يسمى برامج التطبيقات **(Application Program)** وهي برامج مكتوبة بلغة الجافا ويمكن تنفيذها مباشرةً من خلال بيئة الجافا باستخدام مفسر الجافا"Java Interpreter" .

أما النوع الثاني فهو ما يسمى بـ **Applet Program** وهذه البرامج يتم تنفيذها من خلال متصفحات الانترنت مثل Internet Explorer أو Netscape Navigator أو غيرها من متصفحات الانترنت وبالتالي يمكن تنفيذ هذه البرامج على أي حاسب ومع أي متصفح للانترنت وهذا ما يؤكد خاصية الحمل (النقل) لبرامج الجافا أي إمكانية تنفيذها على حاسب يدعم متصفحات الانترنت.

وسوف نستعرض في هذا الفصل برامج التطبيقات أما بالنسبة لبرامج Applet كيف يمكن كتابتها وترجمتها وتنفيذها سواء من خلال بيئة العمل أو من خلال المتصفحات؟، انظر ملحق "ب".

وكما ذكرنا سابقاً أننا سوف نتعرف على مكونات الجافا من خلال أمثلة وذلك بشرح هذه الأمثلة والتعليق عليها سطرا سطرا.

مثال١

اكتب برنامجاً تطبيقياً بسيطاً بلغة الجافا يطبع العبارة التالية

Welcome to Java Programming!

الحل :

البرنامج مبين في شكل (1-2)

```
1.   // Fig. 2-1: Welcome1.java
2.   // A first program in Java.
3.
4.       public class Welcome1 {
5.   // main method begins execution of Java application
6.
7.    public static void main( String args [  ] )
8.   {
9.   System.out.println( "Welcome to Java Programming!" );
10.
11.  }  // end method main
12.
13.  }  // end class Welcome1
```

خرج البرنامج Welcome to Java Programming!

شكل رقم (1-2) البرنامج الأول والخرج الناتج منه

البرنامج المبين في شكل (1-2) برنامج يطبع عبارة الترحيب Welcome to Java Programming!

ومن دراسة هذا البرنامج يتضح الآتي: -

1- إن بعض الحروف كتبت صغيرة small والبعض الآخر كتبت كبيرة capital وهذا يعني أن الحروف الكبيرة تختلف عن الحروف الصغيرة بالنسبة للمترجم، ولذلك ينبغي أخذ الحيطة والحذر الشديد عند كتابة البرنامج والتقيد بالكتابة بالحروف الكبيرة أو الصغيرة عند استخدام أسماء المتغيرات وغيرها ، فمثلاً الحاسب يفرق بين كلٍ من الاسمين التاليين وهما مختلفان sum, Sum لأن أحدهما يبدأ بحرف كبير والآخر يبدأ بحرف صغير ولذلك فإن المترجم يعاملهما مختلفين. فالجافا تعتبر من اللغات الحساسة لحالة الحرف أي لا تتساوى فيها الحروف الكبيرة Capital Letters والحروف الصغيرة Small Letters .

2- إن كل البرامج التي ستذكر في هذه الحقيبة سوف يتم وضع رقم للسطر حتى وإن كان خالياً لا يحتوي على شيء وذلك لسهولة التعليق عليها وسهولة الإشارة إليها ويجب أن نعلم أن هذه الأرقام ليست جزءاً من برنامج الجافا ولا يجب كتابتها عند كتابة البرنامج.

شرح البرنامج

السطر الأول **// Fig. 2-1: Welcome1.java**

جملة من جمل التعليق.

جملة التعليق Comment Statement

تبدأ بـ// ثم يأتي بعدها أي نص مثل سطر١ ، سطر٢ ، وجمل التعليق يتم إهمالها أثناء ترجمة البرنامج وتنفيذه فهي جملة غير تنفيذية.

وتستخدم جمل التعليق لشرح البرنامج وتوثيقة داخلياً وكذلك للتعريف بوظيفة كل جزء وهي تسهل قراءة البرنامج وتعطي فكرة عن وظيفة كل جزء فيه عند كتابتها. وجمل التعليق قد تأتي في سطر واحد فقط أو جزء من سطر وفي هذه الحالة يجب أن تسبق بـ // أما إذا زادت جملة التعليق عن سطر فإنه في هذه الحالة يتم استخدام * / delimiter بحيث تبدأ بها الجمل وتنتهي بـ * / delimiter.

مثال على ذلك

```
/* This is a multiplier line
comment it can be split
into several lines */
```

وكل الجمل بين /*........... */ يتم إهمالها بواسطة المترجم Compiler، وجمل التعليق تفيد المبرمج في أنها تتيح له الفرصة لإضافة أي شرح لأي جزء من أجزاء البرنامج، ويمكن كتابة جمل التعليق بين العلامتين و / * و **/ وفي هذه الحالة يمكن استخدام خاصية من خصائص البرمجة بلغة الجافا وهي Javadoc لكي تقوم بقراءة البرنامج وتجميع كل التعليقات الموجودة فيه لعمل توثيق كامل للبرنامج ولكننا لن نتعرض لهذه الخاصية لأنها خارج نطاق هذه الحقيبة.

السطر الثاني **// A first program in Java**

جملة تعليق ثانيه تبين الغرض من البرنامج .

السطر الثالث سطر فارغ – المبرمج يستخدم الأسطر الفارغة والفراغات البينية لكي يُسهل قراءة البرنامج، والأسطر الفارغة والمسافات الفارغة تُهمل بواسطة المترجم ويمكن استخدامها وقتما يشاء المبرمج.

السطر الرابع

public class Welcome1 {

وهو يبدأ بتعريف الكائن class وإعطائه اسم (identifier). كل برنامج بلغة جافا يحتوي على الأقل على تعريف لكائن واحد يقوم المبرمج بتعريفه. وهذه الكائنات هي الكائنات المعرفة عن طريق المستخدم User defined classes.

والكلمة class تقوم بتعريف الكائن ويتبعها اسم هذا الكائن وهو Welcome1 (في هذا البرنامج). والكلمة class من الكلمات المحجوزة في اللغة التي لها استخدامات خاصة ولذلك لا تصلح لأن تستخدم كاسم معرفي (identifier).

وعند كتابة أسماء الكائنات يفضل أن يبدأ الحرف الأول في اسم class بحرف كبير مثل Welcome1. .وكذلك إذا كان يتكون من أكثر من اسم فإن كل اسم يبدأ بحرف كبير مثل SampleClassName واسم الكائن يعرف بالاسم المعرفي identifier .

الاسم المعرفي identifier

يتكون الاسم المعرفي من مجموعة من الحروف (a-z, A-Z) والأرقام (9 → 0) بالإضافة إلى _ ، $

ويجب أن يراعى عند اختيار الاسم ما يلي : -

١ - أن يبدأ الاسم بحرف.

٢ - أن لا يبدأ برقم .

٣ - لا يحتوي على مسافة فارغة.

٤ - لا يكون من الأسماء المحجوزة (راجع قائمة الأسماء المحجوزة بشكل(2-2)).

٥ - يفضل أن يكون اسماً معبراً عن ما يقوم به الكائن.

٦ - لا يحتوي على أي حروف أو علامات خاصة أخرى غير المذكورة سابقاً.

ومن الأمثلة على ذلك

Welcome1, $Value, _Value…….S_ identified, ………… etc

ومن الأمثلة الخاطئة للاسم المعرفي ما يلي: -

a) 7button — لأنه يبدأ برقم.

b) Input filed1 — لأنه يحتوي على مسافة.

c) Sum+total — يحتوي على "+".

d) public — كلمة محجوزة

Java Keywords الكلمات المحجوزة في لغة الجافا		
abstract	finally	public
boolean	float	return
break	for	short
byte	if	static
case	implements	super
catch	import	switch
char	instanceof	synchronized
class	int	this
continue	interface	throw
default	long	throws
do	native	transient
double	new	true
else	null	try
extends	package	void
false	private	volatile
final	protected	while

شكل (2-2) الكلمات المحجوزة في لغة الجافا

ونذكر مرة أخرى بأن الحروف الكبيرة لا تساوي الحروف الصغيرة في لغة الجافا. وخلال هذه الحقيبة عند تعريف الكائن (class) يجب أن يبدأ بكلمة public. وعند حفظ البرنامج في ملف يجب أن يكون اسم الملف هو نفسه اسم الكائن class متبوعاً بـ ".java" وكل الكائنات المعرفة عن طريق المستخدم يجب أن تحفظ في ملفات لها الامتداد ".java" وسوف يعطي المترجم خطأ عند الترجمة إذا لم يكن اسم الملف هو نفس اسم الكائن. وكذلك إذا لم يكن امتداد الملف java.

والقوس الأيسر في نهاية السطر الرابع } يبين بداية تعريف الكائن (class) ويجب أن ينتهي الكائن (class) بالقوس الأيمن { كما في السطر الثالث عشر من البرنامج.

خطأ شائع:

١. حفظ البرنامج في ملف باسم مختلف عن اسم الكائن (class) حتى ولو كانت نفس الحروف ولكنها تختلف عنها في الحروف الصغيرة والكبيرة يعطي عبارة خطأ عند الترجمة.

٢. استخدام امتداد للملف غير الامتداد المطلوب وهو java. يعطي أيضاً عبارة خطأ عند الترجمة.

ملحوظة : الأسطر التي تمثل جسم الكائن يفضل إزاحتها إلى اليمين قليلاً وذلك لتسهيل القراءة وتسهيل متابعة البرنامج وهذه الإزاحة تهمل عند الترجمة بواسطة المترجم أو المفسر .

السطر الخامس

// main method begins execution of Java application

هو جملة تعليق تبين الغرض من الأسطر 6-11 من البرنامج في شكل (١)

السطر السادس سطر فارغ لتسهيل القراءة

ملاحظة : يمكن إضافة سطر فارغ من الكتابة في أي مكان وذلك لتيسير القراءة

السطر السابع

public static void main(String args[ ])

يمثل جزءاً من كل تطبيق جافا (Java Application) حيث يبدأ تنفيذ البرنامج من الـ main ، والأقواس بعد الـ main توضح أن الـ main هو أحد المقاطع الرئيسة (block) في بناء التطبيق ويسمى method(الطريقة). وكل كائن (class) يجب أن يحتوي على الأقل على طريقة (method) واحدة وقد يحتوي على أكثر من طريقة ويجب أن تكون واحدة من هذه الطرق على الأقل تسمى main ويجب أن تعرف كما في السطر السابع. وفي حالة عدم وجود الmain فإنه لن يتم تنفيذ أي جزء من أجزاء البرنامج. والطرق (methods) تقوم بمعالجة البيانات وأداء بعض العمليات وبالتالي ينتج عنها بعض البيانات أو الخرج عند اكتمال تنفيذها.

والكلمة المحجوزة void تبين أن (الطريقة) method سوف تقوم بأداء عملية ما مثل (طباعة سطر - حساب مضروب عدد ما – حساب المتوسط الحسابي .... الخ)

في هذه الحقيبة سيكون السطر السابع هو أول سطر في بداية method main وسوف يأخذ نفس الشكل الموجود في السطر السابع.

السطر الثامن القوس الأيسر} يحدد بداية method main.

بينما السطر الحادي عشر القوس الأيمن { يحدد نهاية method main .

وكما تم إزاحة الأسطر التي تُكَّوِن الـ class فإنه لتسهيل القراءة والبرمجة فكذلك يمكن إزاحة الأسطر التي تمثل جسم الـ main إلى اليمين قليلاً لنفس السبب.

السطر التاسع

System.out.println("Welcome to Java Programming!");

يخبر الكمبيوتر بطباعة الجملة !Welcome to Java Programming والموجودة بين علامات التنصيص " ". والجملة بين علامات التنصيص تسمى String والمسافات الفارغة في String تهمل بواسطة المترجم .

الجملة System.out تعرف بأنها جملة الخرج القياسية Standard Output Object، .وهذه الجملة تقوم بإظهار الجمل النصية وكذلك أي معلومات أو بيانات في نافذة الأوامر حيث يتم تنفيذ برامج الجافا. والـ Method المسماة System.out.println في هذا البرنامج تظهر النص في سطر واحد في نافذة الأوامر Command Window وعندما تنتهي الطباعة فإن المؤشر يوضع في بداية السطر التالي، وهذا يماثل ضغط مفتاح Enter في لوحة المفاتيح عند الكتابة.

وفي نهاية السطر وضعت الفاصلة المنقوطة ; وهذا يعني أن جملة جافا (Java Statement) قد انتهت. وكل جملة من جمل الجافا يجب أن تنتهي بفاصلة منقوطة. والفاصلة المنقوطة تحدد نهاية الجملةStatement Terminal.

خطأ شائع:

عدم وضع فاصلة منقوطة في نهاية جملة الجافا (أحياناً تكتب الجملة على أكثر من سطر) فإن المترجم يعتبر الجملة غير منتهية ويعطي خطأ.

بعض المبرمجين يجد صعوبة أو عدم وضوح عند استخدام الأقواس ولذلك فإنهم يفضلون كتابة جملة تعليق توضح هل القوس نهاية class أو method أو غيرهما. كما هو واضح في السطر الحادي عشر الذي ينهي method ولذلك تستخدم جملة تعليق مثل الموجودة في السطر الحادي عشر والثالث عشر.

السطر الحادي عشر **} // end method main**

والسطر الثالث عشر **(class)** **} // end class Welcome1 .**

ترجمة وتنفيذ البرنامج الأول

نحن الآن نستطيع ترجمة وتنفيذ هذا البرنامج. ولترجمة البرنامج فإننا نستطيع عمل ذلك من خلال كتابة الأوامر في نافذة الأوامر أو باستخدام القوائم الموجودة في بيئة التشغيل المستخدمة مع Java مثلForte ,Kawa أو غيرهما . وسوف نستعرض أولاً الترجمة والتنفيذ من خلال نافذة الأوامر ثم بعد ذلك نشرح كيف تتم الترجمة والتنفيذ من خلال بيئة العمل .

أولاً الترجمة والتنفيذ باستخدام نافذة الأوامر

١ - غير الفهرس إلى الفهرس الذي تم حفظ البرنامج فيه ثم اكتب الأمر التالي
Javac Welcome1.Java

إذا كان البرنامج يحتوي على أخطاء بنائية (Syntax Errors) فإن هذه الأخطاء سوف تظهر في نافذة الأوامر موضحاً فيها رقم السطر ومكان الخطأ وتفسير محتمل للخطأ. وفي هذه الحالة يجب تصحيح هذه الأخطاء في البرنامج ثم إعادة هذه الخطوة السابقة ثانية ويتم تكرارها حتى يصبح البرنامج بدون أخطاء ويعطي العبارة التالية No Errors . وفي هذه الحالة فإن المفسر يقوم بإنشاء وحفظ ملف جديد يسمى Welcome1.class يحتوي على الـ Byte code. هذا الملف ينتج من ترجمة جمل لغة الجافا بواسطة المترجم إلى byte code وهي صورة أخرى للبرنامج وهي الصورة التنفيذية للبرنامج. و لتنفيذ البرنامج من خلال نافذة الأوامر نكتب الأمر java Welcome1 في نافذة الأوامر. وإذا لم يتوفر الملف ذو الامتداد class. فإن المفسر لا يستطيع تنفيذ البرنامج ويعطي رسالة خطأ.

وتنفيذ البرنامج يبدأ من main method ثم ينتقل إلى الجمل التنفيذية فيه والتي تقوم بإظهار الجملة بين علامتي التنصيص. وعند تنفيذ البرنامج فإن المفسر يقوم بتنفيذ Byte Code الناتج من عملية الترجمة والموجود في الملف ذي الامتداد class.

ثانيا: تنفيذ البرنامج من خلال بيئة العمل

يتم كتابة وترجمة وتنفيذ البرنامج من خلال بيئة العمل Forte ، انظر ملحق "أ".

تعديل البرنامج الأول

في هذا الجزء سوف يتم شرح مثالين يعتمدان على المثال الأول . الأول منهما يقوم بطباعة النص السابق في سطر واحد باستخدام جملتين، أما الثاني فإنه يقوم بطباعة النص على أكثر من سطر باستخدام جملة Println واحدة.

إظهار سطر نصي باستخدام أكثر من جملة

الجملة Welcome to Java Programming! سوف يتم إظهارها في سطر واحد باستخدام جملتين، والبرنامج الذي يقوم بذلك مبين في شكل(٤).

مثال٢ : برنامج يظهر النص **Welcome to Java Programming!** في سطر واحد باستخدام أكثر من جملة طباعة

```
1. // Fig. 2-3: Welcome2.java
2. // Printing a of text line with multiple statements.
3.
4. public class Welcome2 {
5.
6. // main method begins execution of Java application
7.  public static void main( String args[ ] )
8. {
9.         System.out.print( "Welcome to " );
10.    System.out.println( "Java Programming!" );
11.    }  // end method main
12.    }  // end class Welcome2
```

شكل (2-3) برنامج إظهار سطر باستخدام أكثر من جملة

ومعظم جمل هذا البرنامج تشابه البرنامج المبين بشكل (2-1) ولذلك سوف نعلق فقط على الأسطر الجديدة غير المتشابهة.

السطر الثاني **// Printing a of text line with multiple statements.**

جملة تعليق تبين الهدف من البرنامج

السطر الرابع **public class Welcome2 {**

تعريف الكائن وإعطاؤه اسم Welcome2.

السطر التاسع والسطر العاشر من الـ **method main** .

System.out.print("Welcome to ");
System.out.println("Java Programming!");

تظهران سطراً واحداً من النصوص في نافذة الأوامر . الجملة الأولى تظهر النص Welcome to ثم تضع المؤشر في نهاية هذا السطر بينما الجملة الثانية تبدأ من نهاية هذا السطر وتظهر Java Programming! بعد كلمة to وبعد الطباعة تضع المؤشر في بداية السطر التالي.

الفرق بين println, print يظهر بعد كتابة ما بين القوسين ففي حالة print يتم وضع المؤشر في نهاية الجملة التي تمت كتابتها ، أما في حالة println فإنه يتم وضع المؤشر في بداية السطر التالي. ولذلك فإن السطر العاشر سوف يظهر بعد آخر حرف في الجملة الموجودة في السطر التاسع حيث وضع المؤشر بعد انتهاء تنفيذ هذه الجملة.

إظهار عدد من الأسطر باستخدام جملة واحدة

إظهار نص من النصوص في أكثر من سطر واحد باستخدام جملة واحدة يتم ذلك باستخدام ما يسمى حرف السطر الجديد"\n " NewLine Character.

```
1.     // Fig. 2-4: Welcome3.java
2.     // Printing multiple lines with a single statement.
                                                                3.
4. public class Welcome3 {
5.
6.     // main method begins execution of Java application
7.     public static void main( String args[  ] )
8.
9.     {
10.    System.out.println( "Welcome\nto\nJava\nProgramming!" );
11.    }  // end method main
12.    }  // end class Welcome3
```

خرج البرنامج

```
Welcome
to
Java
Programming!
```

شكل (2-4) برنامج طباعة أكثر من سطر باستخدام جملة واحدة وخرجه

وشكل (2-4) يبين البرنامج وكذلك الخرج الناتج من البرنامج حيث يطبع الجملة في أربعة أسطر باستخدام الحرف الخاص بالسطر الجديد" n \"حيث يوضع في المكان المراد بدأ سطر جديد فيه وسوف نشرح الجمل المختلفة عن البرنامج السابق.

مثال ٣ : برنامج يظهر العبارة **Welcome t o Java Programming!** في أربعة أسطر كل كلمة في سطر

// Printing multiple lines with a single statement.

السطر الثاني

تعليق يوضح الهدف من البرنامج وهو طباعة العديد من الأسطر باستخدام جملة واحدة.

السطر الرابع

public class Welcome3 {

تعريف الكائن وإعطاؤه اسم Welcome3 .

السطر العاشر

System.out.println("Welcome\nto\nJava\nProgramming!);

هذه الجملة تظهر أربعة أسطر من النص في نافذة الأوامر. الأحرف المكونة للنص الموضوعة بين علامتي التنصيص يتم إظهارها كما هي بالضبط ولكن الحرفان \ ,n لم يتم طباعتها على الشاشة. والشرطة الخلفية (\ ) تسمى حرف escape وتعتبر من الحروف الخاصة التي تستخدم في جمل الخرج . عند استخدام الشرطة الخلفية في داخل نص فإن الجافا تقوم بضم الحرف التالي للشرطة الخلفية ليكونا معا ما يسمى escape sequence فمثلاً الـ escape sequence \n يعرف بحرف السطر الجديد الذي يحرك المؤشر إلى بداية السطر التالي في نافذة الأوامر.

وشكل (2-5) يوضح بعض الـ escape sequence المعروفة.

الحرف الخاص	الوصف
\n	سطر جديد. يضع المؤشر في بداية السطر التالي
\t	مسافة أفقية. تحريك المؤشر مسافة معينة إلى النقطة التالية في السطر
\r	carriage return. يضع المؤشر في بداية السطر الحالي ولا يتقدم إلى السطر التالي ، وأي حرف يطبع يتم طباعته على حرف سابق تم كتابته في نفس السطر
\\	شرطة خلفية. إظهار "\ " في الخرج
\"	علامة تنصيص مزدوجة. إظهار علامة التنصيص المزدوجة

شكل (2-5) escape sequence

إظهار نص في صندوق الحوار

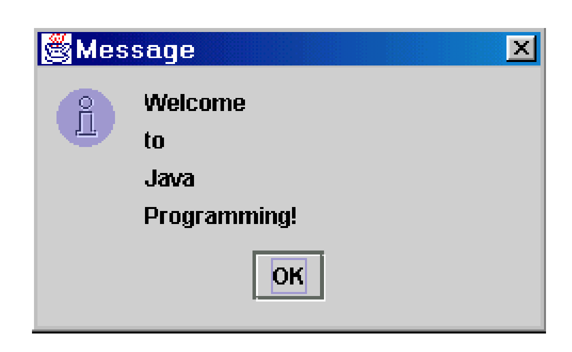
بالرغم من إظهار النص السابق في نافذة الأوامر، إلا أن الكثير من تطبيقات الجافا تستخدم صناديق الحوار لإظهار النصوص بدلاً من نافذة الأوامر، معظم البرامج وبخاصة متصفحات الانترنت مثل Microsoft internet explorer, Netscape Navigator تستخدم صناديق الحوار في كثير من التطبيقات.

وصناديق الحوار هي عبارة عن نافذة يتم إظهار الرسائل المهمة الموجهة للمستخدم فيها، أو التي تعطي خرجاً من البرنامج والـ class المسمى JOptionPane يمدنا بالـ methods التي تساعدنا في إظهار صناديق الحوار المختلفة.

مثال٤

طباعة النص **Welcome to Java Programming!** باستخدام صندوق حوار

```
1. // Fig. 2-6 : Welcome4.java
2. // Printing multiple lines in a dialog box
3.
4. // Java extension packages
5. import javax.swing.JOptionPane;  // import class JOptionPane
6. public class Welcome4 {
7.
8. // main method begins execution of Java application
9. public static void main( String args[  ] )
10.        {
11.        JOptionPane.showMessageDialog(
12.        null, "Welcome\nto\nJava\nProgramming!" );
13.
14.        System.exit( 0 );  // terminate application
15.        }  // end method main
16.        }  // end class Welcome4
```



شكل(2-6) كود البرنامج وصندوق الحوار لإظهار نص في أكثر من سطر

وشكل(2-6) يظهر نفس النص السابق في صندوق حوار يسمى message Dialog. وأحد عناصر القوة في لغة الجافا هو احتواؤها على العديد من الكائنات الجاهزة التي يمكن للمبرمجين إعادة استخدامها ثانيةً بدلاً من إنشائها من البداية.وهذه الكائنات الجاهزة والمحددة مسبقاً يتم تنظيمها وتجميع المتعلق بعضه ببعض في حزم (packages) وهذه الحزم عبارة عن مجموعة من الكائنات (classes) التي تكوّن مكتبة الجافا أو أنها تعرف بما يسمى بـ
java Application Programming Interface (java API). والجافا API تنقسم إلى قسمين أساسيين هما الحزم الأساسية (Core packages) وحزم الامتداد (Extension packages). اسم المجموعة الأساسية يبدأ بـ java بينما الامتداد يبدأ بـ javax. و كثير من هذه الحزم الآن تم دمجها في برنامج الجافا بدءاً من الإصدار الثاني. ويجدر الإشارة هنا إلى أنه نتيجة التطوير المستمر للغة الجافا فإنه يضاف إليها حزم جديدة يمكن تعريفها وتحميلها من الموقع Java.sun.com .
في البرنامج الموضح بشكل (2-6) تم استخدام الكائن JOptionPane الذي تم تعريفه ووضعه في الحزمة javax.swing

السطر الرابع **// Java extension packages**

جملة تعليق في سطر واحد تبين الجزء من البرنامج الذي يشير إلى استدعاء كائن معين من حزمة الامتداد باستخدام import.

وسوف تنقسم جمل import إلى مجموعات: -

١. جمل import للحزم الأساسية (Java).
٢. جمل import لحزم الامتداد (Javax).
٣. جمل import للحزم الخاصة بـDeitel وسوف يتم تعريفها في حينها.

السطر الخامس

```
import javax.swing.JOptionPane;  // import class JOptionPane
```

يوضح جملة import ، والمترجم يستخدم جملة import لكي يتم تعريف وتحميل الكائنات المستخدمة في لغة الجافا ، وفي هذه الحالة استدعاء وتضمين للكائن المسمى JOptionPane.

وعند استخدام الكائنات من API فإن المترجم يتأكد من استخدامها بالصورة الصحيحة. وجملة import تساعد المترجم في أن يحدد ويجد الكائن ولذلك فإنه عند استخدام أي كائن من الجافا API يجب تعريف الحزمة التي تحتوي على هذا الكائن.

يمكنك معرفة بعض المعلومات عن الحزم و الكائنات الموجودة في الجافا API من الموقع Java . sun . com / j2se/1.3/docs/api/index.html كما يمكنك تنزيل الوثائق الخاصة بذلك على حسابك الشخصي من الموقع Java.sun.com / j2se/1.3/docs.html

خطأ شائع:

عدم الالتزام بنفس جملة import كما وردت في السطر الخامس من حيث الأحرف الكبيرة والصغيرة.

في السطر الخامس يخبر المترجم بأن يُحمل الكائن المسمى JOptionPane من الحزمة المسماة javax.swing . وهذه الحزمة تحتوى على كثير من الكائنات الأخرى مثل الكائنات الخاصة بالرسومات والتعامل مع المستخدم من خلال بيئة الرسومات GUI graphical user interface التي تسهل إدخال و إخراج البيانات من خلال مربعات حوار.

السطر الحادي عشر والثاني عشر

```
JOptionPane.showMessageDialog(
 null, "Welcome\nto\nJava\nProgramming!" );
```

يشير إلى استدعاء الـ method المسماة show.Message.Dialog من الكائن المسمى JOptionPane.وهذه الـ method تتطلب مدخلين (two arguments) مفصولين بفاصلة "،". المدخل الأول (first argument) دائماً سيكون الكلمة "null " ، وهو يحدد المكان الذي يظهر فيه صندوق الحوار ، وفي هذه الحالة (استخدام الكلمة "null ") فإن صندوق الحوار سوف يظهر في منتصف الشاشة ، أما المدخل الثاني second argument فهو النص المراد إظهاره.

والـ method المسماة showMessage هي method خاصة من الـكائن المسمى JOptionPane تسمى Static Method. وهذه الـ method دائماً تستدعى باستخدام اسم الـكائن متبوعاً بنقطة يليها اسم الـ method كما هو واضح في الشكل التالي

class name . method name (arguments)

تنفيذ السطرين 11-12 سوف يظهر صندوق الحوار المبين بشكل (2-7).
العنوان الذي يظهر في صندوق الحوار يحتوي على الكلمة Message يبين أن هذا الصندوق يظهر رسالة إلى المستخدم. وصندوق الحوار يحتوي على زر OK يسمح للمستخدم بإخفاء هذا الصندوق بالضغط على الزر OK.



شكل (2-7) صندوق حوار اظهار رسالة

ويجب أن تتذكر أن أي جملة داخل الـ method يجب أن تنتهي بفاصلة منقوطة. لغة الجافا تسمح بتقسيم الجمل الكبيرة على أكثر من سطر وفي هذه الحالة فإن الفاصلة المنقوطة تأتي في نهاية الجملة وليس في نهاية كل سطر. بعض الجمل لا يمكن تقسيمها في أماكن معينة منها، فمثلاً لا يمكن تقسيم الجملة في وسط الأسماء المعرفية(Identifier) أو في وسط أي نص.

السطر الرابع عشر **System.exit(0); // terminate application**

وهذه الجملة تشير إلى استخدام ثابت للـ method المسماة exit والموجودة في الكائن المسمى System وذلك لإنهاء التطبيق. ويجب استخدام هذه الجملة في كل التطبيقات التي تستخدم GUI وذلك لإنهاء التطبيق. ونلاحظ أنه في السطر الرابع عشر استخدمت نفس القاعدة السابقة في استدعاء الـ method من داخل الكائن بكتابة اسم الكائن ثم نقطة ثم يلي ذلك اسم الـ method . الأسماء المعرفية للكائن تبدأ دائماً بحرف كبير Capital. الـكائن system لم يتم استيراده أو دمجه مع

البرنامج لأنه جزء من الحزمة Java.lang، والحزمة Java.lang هي الحزمة الوحيدة التي لا يتطلب استخدام أيٍ من الـ methods المدمجة فيهاويتم استدعاؤها عن طريق جملةimport.
المدخل (argument) (0)للـ method exit يبين أن التطبيق تم إنهاؤه بنجاح وبدون أخطاء. وإذا كان المدخل لا يساوي الصفر فإن ذلك يعني وجود خطأ.

مثال٥

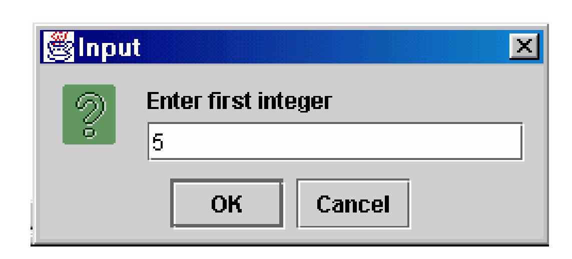
إضافة أرقام صحيحة

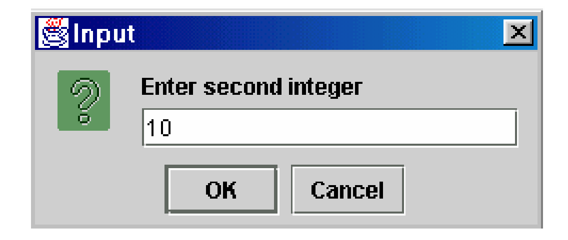
البرنامج التالي يقوم بقراءة رقمين صحيحين من خلال لوحة المفاتيح ثم يحسب مجموعهما ومن ثم يقوم بطباعة مجموع الرقمين

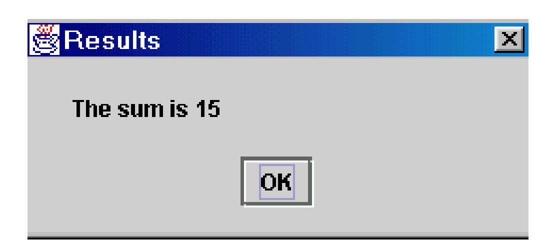
البرنامج وشاشات الإخراج والإدخال موضحة في شكل (8-2)

```
1. // Fig. 2-8 : Addition.java
2. // An addition program.
3.
4. // Java extension packages
5.  import javax.swing.JOptionPane;  // import class JOptionPane
6.
7.  public class Addition {
8.
9.  // main method begins execution of Java application
10.        public static void main( String args[ ] )
11.        {
12.        String firstNumber;   // first string entered by user
13.        String secondNumber;  // second string entered by user
14.        int number1;          // first number to add
15.        int number2;          // second number to add
16.        int sum;              // sum of number1 and number2
17.
18.        // read in first number from user as a string
19.        firstNumber =
20.        JOptionPane.showInputDialog( "Enter first integer" );
21.
```

```
22.        // read in second number from user as a string
23.        secondNumber =
24.        JOptionPane.showInputDialog( "Enter second integer" );
25.        // convert numbers from type String to type int
26.        number1 = Integer.parseInt( firstNumber );
27.        number2 = Integer.parseInt( secondNumber );
28.
29.        // add the numbers
30.        sum = number1 + number2;
31.
32.        // display the results
33.        JOptionPane.showMessageDialog(
34.        null, "The sum is " + sum, "Results",
35.        JOptionPane.PLAIN_MESSAGE );
36.
37.        System.exit( 0 );   // terminate application
38.
39.        }  // end method main
40.
41.        }  // end class Addition
```







شكل(2-8) برنامج إضافة رقمين مع شاشات إدخال وإخراج البيانات

شرح البرنامج

السطران الأول والثاني

```
// Fig. 2-8 : Addition.java
// An addition program.
```

تمثلان جملة تعليق توضح الشكل الذي يظهر فيه البرنامج وكذلك عمل البرنامج .

السطر الرابع يمثل جملة تعليق تبين أن السطر التالي يحتوي على جملة import التي تتضمن استدعاء (تضمين)الحزمة الممتدة للجافا مع البرنامج .

السطر الخامس

```
import javax.swing.JOptionPane;  // import class JOptionPane
```

تقوم هذه الجملة بإخبار المترجم بتحميل الكائن المسمى JOptionPane الموجود في الحزمة javax.swing للاستخدام في البرنامج. وكما ذكر من قبل فإن كل برنامج يجب أن يتكون من كائن واحد على الأقل كما هو مبين في

السطر السابع **public class Addition {**

ولذلك يجب أن يتم حفظ هذا التطبيق (البرنامج) في ملف له نفس اسم الكائن class وهو Addition ولذلك فإن اسم الملف يصبح Addition.java .

ويبدأ الكائن كما هو معروف بالقوس الأيسر } كما هو في السطر السابع وينتهي بالقوس الأيمن { كما في السطر ٤١. وكما هو معروف فإن كل تطبيق يبدأ التنفيذ بالـ method main (الأسطر من ١٠: ٤٠) والتي تبدأ من السطر ١١ الذي يحتوي على القوس الأيسر وتنتهي عند السطر ٣٩ القوس الأيمن الذي يبين نهاية main.

السطر الثاني عشر والسطر الثالث عشر

```
String firstNumber;   // first string entered by user
 String secondNumber;  // second string entered by user
```

هذه الجمل تعرف بجمل التعريف declaration statements.

جمل التعريف declaration statements

هذه الجمل تقوم بتعريف المتغيرات Variables التي سوف تستخدم في داخل البرنامج بتحديد اسم لها وكذلك تحديد نوع البيانات التي ستخزن فيها، ويجب أن تأتي جمل التعريف في بداية الطريقة method لأن لغة الجافا لا تسمح باستخدام أي من المتغيرات إلا بعد تعريفها و أي متغير يستخدم خلال البرنامج يجب أن يكون له اسم وأسماء المتغيرات تتبع نفس القواعد التي ذكرت عند الحديث عن الأسماء المعرفية identifiers ونُذَّكر بهذه القواعد مرةً ثانية وهي: -

- يتكون الاسم من حروف الهجاء a-z ، A-Z والأرقام ($0 \rightarrow 9$) ، $ ، _
- أن لا يبدأ الاسم برقم
- لا يحتوي على مسافات فارغة
- لا يكون من الكلمات المحجوزة
- لا يحتوي على علامات خاصة غير $ ، _

والمتغير يمثل مكاناً في ذاكرة الجهاز حيث سيتم تخزين فيه المتغير في هذا المكان بواسطة البرنامج. ففي هذا البرنامج الكلمات firstNumber و secondNumber هي أسماء متغيرات من النوع String (الموجودة في الحزمة Java.lang) وهذا يعني أن هذه المتغيرات سوف تحمل بيانات من نوع String. كل جملة تعريف يجب أن تنتهي بفاصلة منقوطة " ; "، وقد يضاف في نهايتها (بعد الفاصلة المنقوطة) جملة تعليق لبيان وتوضيح ما يحمله هذا المتغير، وهذه عادة المبرمجين لتوضيح الغرض من هذه المتغيرات حتى يسهل فهم البرنامج لمن يقرؤه وكذلك تعتبر طريقة لتوثيق البرنامج.

يمكن استخدام جملة تعريف لكل متغير كما هو مبين في السطرين الثاني والثالث عشر، كما يمكننا استخدام جملة واحدة لتعريف المتغيرات من نفس النوع وفي هذه الحالة يتم فصل المتغيرات بفاصلة. فالسطران الثاني عشر والثالث عشر يمكن كتابتهما في جملة تعريف واحدة كما يلي

```
String firstNumber,   // first string entered by user
secondNumber;  // second string entered by user
```

الأسطر من ١٤ - ١٦

```
14      int number1;      // first number to add
15      int number2;      // second number to add
16      int sum;        // and sum of number1 numbe
```

هذه الأسطر تعرف المتغيرات number1, number2 and sum بأنها بيانات من النوع int وهذا يعني أن هذه المتغيرات سيوضع بها قيمة عددية صحيحة مثل(7 ,11 ,200 ....الخ). وشكل (2-9) يبين الأنواع الأساسية للبيانات Primitive data type في لغة الجافا والحجم الذي يشغله هذا النوع بالبت (bits). وهذه الأنواع تعتبر من الكلمات المحجوزة في اللغة وتكتب دائماً في بالحروف الصغيرة.

النوع (type)	الحجم بالبت (Size in bits)	المدى (range) أو القيمة (value)	ملاحظات
boolean	١	True or false	قيمة منطقية
char	16	to FFFF٠٠٠٠	متغير يحمل حرفاً واحداً فقط
byte	8	-128 to +127	
short	16	-32,768 to +32767	قيمة صحيحة في المدى الموضح قرين كل نوع
int	32	-2,147,483,648 to +2,147,483,647	
long	64	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807	
float	32	-3.40292347 E+38 to + 3.40292347 E+38	قيمة تحتوي على نقطة عشرية في المدى الموضح قرين كل نوع
double	64	-1.79769313488231570 E+308 to +1.79769313488231570 E+308	

شكل (2-9) أنواع البيانات الأساسية في لغة الجافا

السطر الثامن عشر

```
18     // read in first number from user as a string
```

يبين جملة تعليق توضح أن الجملة التالية ستقوم بقراءة الرقم الأول المدخل بواسطة المستخدم.

السطر التاسع عشر والعشرون

```
19  firstNumber =
20  JOptionPane.showInputDialog( "Enter first integer" );
```

هذه الجملة تقوم بقراءة String يمثل الرقم الأول الذي يجب أن يضاف والـmethod JOptionPane.showInputDialog تظهر مربع حوار لإدخال الرقم كما هو مبين بشكل (2-10).



شكل رقم (2-10) مربع حوار لإدخال قيمة

المدخل (argument) إلى مربع الحوار showInputDialog هو عبارة عن جملة توضح للمستخدم ما يجب عليه فعله وهذه الرسالة تسمى prompt لأنها ترشد المستخدم لعمل فعل معين. ويقوم المستخدم بإدخال البيانات النصية ثم يضغط OK أو يضغط Enter وعندها يتم حفظ المدخلات كنص في المتغير المسمى firstNumber الذي تم تعريفه مسبقا على أنه متغير نصي.

السطر الثاني والعشرون

```
22     // read in second number from user as a string
```

جملة تعليق تبين الغرض من السطرين التاليين وهو إدخال القيمة الثانية التي ستضاف والتي يتم إدخالها كنص

السطران ٢٣ -٢٤

```
23  secondNumber =
24              JOptionPane.showInputDialog( "Enter second integer" );
```

يظهران صندوق حوار لإدخال القيمة الثانية والتي تدخل كنص ويتم حفظها في المتغير المسمى secondNumber

السطر ٢٥

```
// convert numbers from type String to type int
```

جملة تعليق من سطر واحد تبين وظيفة السطرين التاليين وهي تحويل الرقمين السابقين من الصورة النصية إلى الصورة الرقمية.

السطران ٢٦ -٢٧

```
26   number1 = Integer.parseInt( firstNumber );
27   number2 = Integer.parseInt( secondNumber )
```

هاتان الجملتان تقومان بتحويل القيمة النصية المدخلة بواسطة المستخدم والتي تم حفظها في المتغيرين firstNumber, secondNumber إلى قيمة صحيحة حتى نستطيع استخدامها في حساب المجموع.

وال method المسماة Integer.parseInt (هي method ثابتة من الكائن Integer) تحول المدخل النصي إلى قيمة صحيحة. والكائن Integer معرف في الحزمة java.lang وبالتالي لا يتطلب استخدامه استدعاءه بجملة import. وبعد التحويل يتم حفظ القيمة المحولة في المتغيرين number1, number2 على التوالي.

لتجويل متغير نصي إلى متغير من النوع Double فإننا نستخدم الجملة التالية:

variable1= Double.parseDouble (*variable2*)

حيث variable1, variable2 متغيران من النوع double , String على التوالي.

السطر ٢٩

// add the numbers

جملة تعليق من سطر واحد لبيان الغرض من السطر القادم

السطر ٣٠

٢٩ sum = number1 + number2;

هذا السطر يقوم بحساب مجموع المتغيرين number1, number2 وذلك باستخدام عامل الجمع "+" ويحفظ الناتج في المتغير sum باستخدام عامل الإسناد "=" . جميع العمليات الحسابية تتم باستخدام جمل الإسناد، في عمليات الجمع تضاف القيم المخزنة في المتغيرات بعضها إلى بعض. ففي هذا المثال تضاف القيمة الموجودة في المتغير number1 إلى القيمة الموجودة في المتغير number2 ويحفظ الناتج في المتغير sum. ويتضح أيضا من هذا المثال أن عامل الجمع (+) هو عامل ثنائي المدخلات : أي يحتاج إلى معاملين وهما كما في هذا المثال number1, number2 وسوف نتعرض بالشرح لجميع العمليات الحسابية والمنطقية وغيرها من العمليات بالتفصيل في الجزء المتبقي من هذه الوحدة بعد الانتهاء من شرح هذا المثال.

الأسطر ٣٤ -٣٦

```
٣٤        JOptionPane.showMessageDialog(
٣٥        null, "The sum is " + sum, "Results",
٣٦        JOptionPane.PLAIN_MESSAGE );
```

تمثل جملة الخرج والتي تستخدم مربع الحوار لإظهار ناتج عملية الجمع
وهذه النسخة لـ method **JOptionPane.showMessageDialog** تتطلب أربعة مدخلات (Four arguments) هي: -

١ - المدخل الأول null وهو يخبر الحاسب بوضع صندوق الحوار في وسط الشاشة.

٢ - المدخل الثاني هو الرسالة التي سوف تظهر في صندوق الحوار وفي هذا المثال هي "The sum is " + sum وهنا تم استخدام العامل " + " لإلحاق القيمة المُخزَّنة في المتغير sum بعد تحويلها إلى نص في نهاية النص "The sum is ".

ويجب التفريق بين استخدام العامل " + " كعامل إضافة للقيمة الحسابية وعامل إلحاق بين النصوص في جملة الخرج (مربع الحوار).

٣ - المدخل الثالث يمثل النص الذي سوف يظهر في سطر العنوان لمربع الحوار، في هذا المثال "Results" (راجع شكل (2-8) السابق.

٤ - المدخل الرابع JOptionPane.PLAIN_MESSAGE يُمثل الرمز الذي يبين نوع مربع الحوار.

ويوجد مجموعة من الرموز التي يمكن اظهارها في صندوق الحوار لتساعد المستخدم في معرفة نوع صندوق الحوار والرسالة التي تظهر فيه وهذه الرموز مبينة في شكل (2-11)

نوع رسالة صندوق الحوار	الرمز	الوصف
JOptionPane. ERROR_ MESSAGE		عرض صندوق حوار يبين رسالة خطأ للمستخدم
JOptionPane. INFORMATION_ MESSAGE		عرض صندوق حوار مع رسالة للمستخدم
JOptionPane. WARNING_ MESSAGE		رسالة تحذيرية للمستخدم
JOptionPane. QUESTION_ MESSAGE	?	سؤال للمستخدم يجب الإجابة عليه بنعم أو لا
JOptionPane. PLAIN_ MESSAGE	لا يوجد رمز	يظهر رسالة في الصندوق بدون رموز

شكل (2-11) الرموز التي تظهر مع صندوق الحوار

أنواع العمليات

تتعدد العمليات في لغات البرمجة والتي تتيح للمبرمج من تحقيق الهدف من البرنامج وفيما يلي سنعرض أنواع هذه العمليات:

العمليات الإسنادية Assignments

تستخدم هذه العملية لتخصيص قيمة ما في متغير وذلك بعد تعريفه ، ونستخدم العملية = للتعبير عن التخصيص في لغة الجافا :

أمثلة:

x = 1 ;

تم تخصيص 1 إلى المتغير x

radius = 1.0 ;

تم تخصيص القيمة 1.5 إلى المتغير x

a = 'A' ;

تم تخصيص الحرف 'A' إلى المتغير a

خطأ شائع : تخصيص قيمة من نوع بيانات مختلف عن نوع المتغير ، فمثلا إذا كان x=1.0 يمكن أن تعطي خطأ وذلك إذا كان x معرف على أنه قيمة صحيحة int ، لذلك يجب أن يكوناً معرف على أنه double أي يقبل الكسور .

أيضا لاحظ أن اسم المتغير يجب أن يكون على اليسار بمعنى أن الجملة 1 = x تعتبر خطأ .

يمكن أن تحتوي جملة التخصيص على تعبير فمثلا :

area = radius * radius * 3.14159 ;

x = x + 3 ;

يمكن كتابة العملية الإسنادية السابقة بشكل مختصر x + = 3

واستعمال مثل هذه الاختصارات يسهل كثيرا جدا على المبرمجين كما يستطيع المترجم العمل أسرع عند ترجمة البرنامج .

والشكل التالي يوضح الاختصارات الشائعة الاستخدام في العمليات الإسنادية:

العامل	مثال	ما يقابله دون اختصار
+ =	c + = 7	c = c + 7
- =	d - = 4	d = d - 4
* =	e * = 5	e = e * 5
/ =	f = f / 3	f = f / 3
% =	g % = 9	g = g % 9

شكل (12) اختصارات العمليات الإسنادية

عامل الزيادة وعامل النقصان

تمدنا لغة الجافا بعامل الزيادة + + وعامل النقصان - - ويستطيع المبرمج زيادة قيمة متغير بمقدار الوحدة عن طريق استخدام عامل الزيادة + + مثل + + c وذلك بدلا من استخدام أي الجمل التالية

c + = 1; أو c = c + 1;

ويمكن كتابة عامل الزيادة أو عامل النقصان قبل أو بعد المتغير وذلك حسب الحاجة لاستخدامه داخل البرنامج.

والشكل التالي يوضح الفرق بين استخدام هذه العوامل قبل المتغير أو بعده

العامل	مثال	توضيح
+ +	+ + a	يتم زيادة المتغير a بمقدار 1 ثم تستخدم القيمة الجديدة للمتغير a في التعبير المتواجد فيه
+ +	+ + a	تستخدم القيمة الأولى للمتغير a في التعبير المتواجد فيه ثم بعد ذلك يتم زيادة المتغير a بمقدار 1
--	--b	يتم إنقاص المتغير b بمقدار 1 ثم تستخدم القيمة الجديدة للمتغير b في التعبير المتواجد فيه
--	b--	تستخدم القيمة الأولى للمتغير a في التعبير المتواجد فيه ثم يتم إنقاص المتغير b بمقدار ١

شكل (13) عامل الزيادة وعامل النقصان

الشكل التالي يوضح استخدام عامل الزيادة قبل أو بعد المتغير.

```
1.  // Fig. 2.14  Increment.java
2.  // Preincrementing and postincrementing
3.

4.  public class Increment {
5.  public static void main( String args[] )
6.  {
7.  int c;
8.
9.  c = 5;
10. System.out.println( c );   // print 5
11. System.out.println( c++ ); // print 5 then postincrement
12. System.out.println( c );   // print 6
13.
14. System.out.println();      // skip a line
15.
16. c = 5;
17. System.out.println( c );   // print 5
18. System.out.println( ++c ); // preincrement then print 6
19. System.out.println( c );   // print 6
20. }
21. }
```

```
Output Window
5
5
6

5
6
6
Program Input:
```

شكل (2.14) مثال على استخدام عامل الزيادة وعامل النقصان

شرح البرنامج

السطر رقم 10

يتم طباعة قيمة المتغير c وهي 5

```
System.out.println( c );   // print 5
```

السطر رقم 11

```
system . out . println ( c + + ) ;
```

لاحظ أن عامل الزيادة أتى بعد المتغير لذلك يتم استخدام القيمة القديمة للمتغير c وهى 5 وتظهر 5 في جملة الطباعة ثم بعد الطباعة يتم زيادة المتغير c بمقدار ١ ويصبح c = 6

السطر 12

جملة طباعة للمتغير c بقيمة 6

```
System.out.println( c );   // print 6
```

السطر 14

```
System.out.println ( ) ;
```

طباعة سطر خالٍ

السطر 16

```
c = 5 ;
```

إعادة تخصيص القيمة 5 للمتغير c

السطر 17

طباعة قيمة المتغير c وهى 5

```
System.out.println( c );   // print 5
```

السطر 18

```
System.out.println( ++c ); // preincrement then print 6
```

لاحظ أن عامل الزيادة أتى قبل المتغير لذلك تتم أولا عملية الزيادة فيصبح المتغير c يساوي 6 ثم يتم استخدام القيمة الجديدة في جملة الطباعة فيطبع القيمة 6

السطر 19 جملة طباعة للمتغير c بالقيمة 6

```
System.out.println( c );   // print 6
```

العمليات الحسابية

معظم برامج الكمبيوتر تقوم بعمل عمليات حسابية والشكل التالي يوضح بعض العمليات الحسابية

العملية	العامل	التعبير الجبري	التعبير بالجافا
جمع	+	f+7	f+7
طرح	-	f-7	f-7
ضرب	❖	bm	b*m
قسمة	/	x	x/y
موديلاس	٪	r mod s	r%s

شكل (2.15) العمليات الحسابية

لاحظ الفرق بين التعبير الجبري والتعبير بواسطة الجافا وذلك في بعض العمليات مثل عملية الضرب والتي يعبر عنها بالعلامة (❖) ، عملية القسمة (/) ، أيضا عملية الموديلاس (٪) .

ملحوظة عمليات القسمة التي تتم على الأعداد الصحيحة تعطي عدداً صحيحاً فمثلا

خارج قسمة 4 / 7 هو 1

خارج قسمة 5 / 17 هو 3 وهكذا كما ترى يهمل الجزء العشري.

تمدنا الجافا بالمعامل (٪) الذي يعطي الباقي بعد القسمة ولاحظ أن هذه العملية تتم فقط على معاملات صحيحة فمثلا

ناتج العملية 4 % 7 هو 3

ناتج العملية 5 % 17 هو 2

تتعدد استخدامات المعامل (٪) كما سنرى إن شاء الله في الفصول القادمة.

خطأ شائع محاولة استخدام المعامل (٪) مع معاملات غير صحيحة يعطي syntax error

أولوية تنفيذ العمليات الحسابية

تستخدم الأقواس في الجافا تماما كما في الجبر العادي فمثلا

ضرب عدد a في حاصل جمع عددين تكتب هكذا $a * (b+c)$

تقوم الجافا بتنفيذ العمليات الحسابية بترتيب دقيق مشابه لما يحدث في الجبر:

١ - تقوم بحساب ما بداخل الأقواس أولا ، ولذلك تستخدم الأقواس من قبل المبرمج لحساب عملية معينة بترتيب معين يحدده المبرمج حسب ترتيب الأقواس.

٢ - إذا كانت هناك عملية بها عدة أقواس متداخلة يحسب أقصى قوس أولا من الداخل ثم الذي يليه إلي الخارج وهكذا ... .

٣ - يأتي بعد ذلك في الأولوية العمليات (الضرب ، القسمة ، الموديلاس) وهذه العمليات تعتبر في نفس درجة الترتيب. أما إذا كانت العملية الحسابية تحتوي على عدد من عمليات الضرب والقسمة و الموديلاس فإن التنفيذ يبدأ من اليسار إلى اليمين .

٤ - يأتي بعد ذلك الجمع والطرح وهما في نفس درجة الترتيب ولهذا مثل ما سبق إذا وجدت أكثر من عملية جمع وطرح في عملية حسابية واحدة فإن التنفيذ يبدأ من اليسار إلى اليمين .

مثال عبر عن العمليات الحسابية التالية بلغة الجافا وبين ترتيب تنفيذ العمليات داخل

y=mx+b
z= pr % q + w/x
y=x +bx +c

- التعبير بالجافا

```
y=m*x +b;
```

لاحظ عدم وجود أقواس لذلك ينفذ الضرب أولا m*x ثم تنفذ بعد ذلك عملية الجمع .

- التعبير بالجافا

```
z = p *  r  %  q  +  w / x - y  ;
      6   1   2    4   3   5
```

الأرقام السابقة تمثل ترتيب تنفيذ العمليات

لاحظ أن الضرب والقسمة و الموديلاس لها نفس مستوى الترتيب ولهذا يكون الترتيب من اليسار لليمين

-التعبير بالجافا

```
y = a * x * x  + b * x  + c
  6   1   2    4   3    5
```

في المثال السابق بفرض القيم التالية :

a = 2 , b = 3 , c = 7 , x = 5

y = 2 * 5 * 5 + 3 * 5 + 7;

الخطوة الأولى 2 * 5 = 10

y = 10 * 5 + 3 * 5 + 7;

الخطوة الثانية 10 * 5 = 50

y = 50 + 3 * 5 + 7;

الخطوة الثالثة 3 * 5 = 15

y = 50 + 15 + 7;

الخطوة الرابعة 50 + 15 = 65

y = 65 + 7;

الخطوة الخامسة 65 + 7 = 72

الخطوة السادسة y = 72;

كما في التعبير الجبري من المفضل استعمال الأقواس حتى غير الضرورية وذلك لجعل التعبير أوضح

Y = (a * x * x) + (b * x) + c ;

العاملات المنطقية

يدرج الجدول التالي المعاملات المنطقية شائعة الاستخدام في برمجة java

العامل	العملية
&&	And المنطقية
||	Or المنطقية
!	Not المنطقية

شكل (2.16) العاملات المنطقية

يتم استخدام العاملات المنطقية مع المعاملات التي يكون لها إحدى قيم true أو false، أو التي تكون قيم يمكن تحويلها إلى true أو false .

يقوم العامل المنطقي && بتقييم اثنين من المعاملات وإرجاع القيمة true إذا كان كلا المعاملين true . فيما عدا ذلك ، سيرجع العامل && القيمة false .

يتم استخدام ذلك في التفرع الشرطي حيث يتم تحديد اتجاه برنامج java عن طريق اختبار اثنين من الشروط .إذا تم تلبية كلا الشرطين ، سيتوجه البرنامج إلى اتجاه معين ، فيما عدا ذلك ، سيأخذ البرنامج اتجاه مختلف .

على العكس من العامل && الذي يحتاج كلا المعاملين لكي يكون true ، يقوم العامل || بتقييم معامليه ويرجع القيمة true . اذا لم يرجع أي من المعاملين القيمة true ،فإن العامل سيرجع القيمة false يمكن الاستفادة من ذلك في برمجة java لأداء إجراء معين إذا لم تتم تلبية أيا من شرطي الاختبار .

يعتبر العامل المنطقي الثالث ! عامل أحادي يتم استخدامه قبل معامل واحد . يقوم هذا العامل بإرجاع القيمة العكسية للمعامل المعطى ، وبذلك ، إذا كان المتغير a له القيمة true ، فإن a! سيكون له القيمة false . في برنامج java ، من الأفضل تبديل قيمة المتغير في التكرارات المتتالية للحلقة باستخدام عبارة مثل a=!a . يؤكد ذلك أنه في كل دورة سيتم تغيير القيمة .

أمثلة على العاملات المنطقية

يوضح البرنامج التالي الكيفية التي يمكن بها اختبار القيم المنطقية باستخدام المعاملات المنطقية التي سبق عرضها :

```
1.  // Fig.  2.17    : Logical.java
2.  // Logical Operator
3.  public class Logical
4.  {
5.  public static void main ( String [] args )
6.  {
7.  //declare & initialize test variable
8.  boolean a  = true , b = false ;
9.  boolean c1 =(a && a);      // test if both are true
10. boolean c2 =(a && b);
11. boolean c3 =(b && b);
12.
13. boolean c4 =(a || a) ;       //test if either is true
14. boolean c5 =(a || b);
15. boolean c6 =(b || b);
16.
17. boolean c7 = !a;             // invert initial values
18. boolean c8 = !b:
19.
20. // display the results
21. System.out.println ("and:\n1:"+c1+"2:"+c2+"3:"+c3);
22. System.out.println ("or:\n4:"+c4+"5:"+c5+"6:"+c6);
23. System.out.println ("not:\n7:"+c7+"8:"+c8);
24. }
25. }
```

```
Output Window
and:
1:true2:false3:false
or:
4:true5:true6:false
not:
7:false8:true

Program Input:
```

شكل(2.17) مثال على العمليات المنطقية

شرح البرنامج

السطر رقم 8 تم تعريف متغيرين من النوع boolean هما a ، b وإعطائهما قيم ابتدائية هي true ، false على الترتيب.

السطور 9,10,11

تم تعريف المتغيرات c1,c2,c3 وذلك لمناقشة الحالات المختلفة للعملية &&

السطور 13,14,15

تم تعريف المتغيرات c4,c5,c6 وذلك لمناقشة الحالات المختلفة للعملية ||

السطور 17,18

تم تعريف المتغيرين c7,c8 وذلك لمناقشة الحالات المختلفة للعملية !

السطور 21,22,23

هي جمل لطباعة المتغيرات c1,c2,c3,c4,c5,c6,c7,c8.

اتخاذ القرار: التساوي والعمليات العلاقية

سوف نناقش في هذا الجزء جملة if في أبسط صورها والتي تسمح للبرنامج بأخذ قرار معين معتمد على صحة أو خطأ شرط ما .

إذا تحقق الشرط نفذت الجملة التالية لجملة if أما إذا لم يتحقق لم تنفذ الجملة التالية لجملة if .

ملحوظة: في الحقيقة جملة **if** والجملة التي تليها هما جملة واحدة.

الشروط في جملة if تكتب باستخدام عمليات التساوي والعمليات العلاقية والجدول التالي يوضح جميع عمليات التساوي والعمليات العلاقية في لغة الجافا:

شكل عمليات التساوي في الجبر	شكل عمليات التساوي في الجافا	مثال على الشرط في الجافا	معنى الشرط
=	==	x==y	x تساوي y
≠	!=	x!=y	x لا تساوي y

شكل (2.18) عمليات التساوي

شكل العمليات العلاقية في الجبر	شكل العمليات العلاقية في الجافا	مثال على الشرط في الجافا	معنى الشرط
>	>	x>y	x أكبر من y
<	<	x<y	x أقل من y
≥	>=	x>=y	x أكبر من أو تساوي y
≤	<=	x<=y	x أقل من أو تساوي y

شكل (2.19) العمليات العلاقية

خطأ شائع:

عند كتابة العمليات =! ، == ، = < ، =< ، وبينهما مسافة مثل == ، => ،=< ، =! يعطي البرنامج syntax error.

أيضا عند عكس رموز العملية الواحدة مثل =< ، => ، =! يعطي أيضا syntax error

المثال التالي نستخدم 6 جمل شرطية if للمقارنة بين رقمين مدخلين بواسطة المستخدم وذلك كتطبيق على عمليات التساوي والعمليات العلاقية.

في هذا المثال يقوم المستخدم بإدخال قيمتين من خلال صندوق حوار ، بعد ذلك يقوم البرنامج بتحويل هاتين القيمتين إلى عددين صحيحين ثم يقوم بتخزين الرقمين في المتغيرين number2 ،number1. ثم يقوم البرنامج بعمل المقارنات عن طريق جمل if ويعرض الناتج في صندوق المعلومة .

الشكل التالي يوضح البرنامج ونموذج لشكل المخرجات.

```
1.  // Fig. 2.20: Comparison.java
2.  // Compare integers using if structures, relational operators
3.  // and equality operators.
4.
5.  // Java extension packages
6.  import javax.swing.JOptionPane;
7.
8.  public class Comparison {
9.
10. // main method begins execution of Java application
11. public static void main( String args[] )
12. {
13. String firstNumber;
14. String secondNumber;
15. String result;
16. int number1;
17. int number2;
18.
19. // read first number from user as a String
20. firstNumber =
21. JOptionPane.showInputDialog( "Enter first integer:" );
22.
23. // read second number from user as a String
24. secondNumber =
25. JOptionPane.showInputDialog( "Enter second integer:" );
26.
27. // convert numbers from type String to type int
28. number1 = Integer.parseInt( firstNumber );
29. number2 = Integer.parseInt( secondNumber );
30.
31. // initialize result to empty String
32.
33. result = "";
34. if ( number1 == number2 )
35. result = number1 + " == " + number2;
36.
37. if ( number1 != number2 )
38. result = number1 + " != " + number2;
39.
```

```
40. if ( number1 < number2 )
41. result = result + "\n" + number1 + " < " + number2;
42.
43. if ( number1 > number2 )
44. result = result + "\n" + number1 + " > " + number2;
45.
46. if ( number1 <= number2 )
47. result = result + "\n" + number1 + " <= " + number2;
48.
49. if ( number1 >= number2 )
50. result = result + "\n" + number1 + " >= " + number2;
51.
52. // Display results
53.
54. JOptionPane.showMessageDialog(
55. null, result, "Comparison Results",
56. JOptionPane.INFORMATION_MESSAGE );
57. System.exit( 0 );  // terminate application
58.
59. }  // end method main
60.
61. }  // end class Comparison
```

Input
Enter first integer:
111
OK Cancel

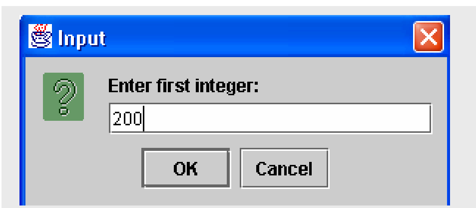
Input
Enter first integer:
111
OK Cancel

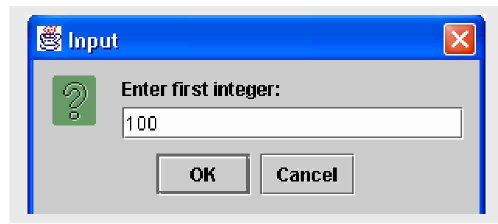
Comparison Results
111 == 111
111 <= 111
111 >= 111
OK

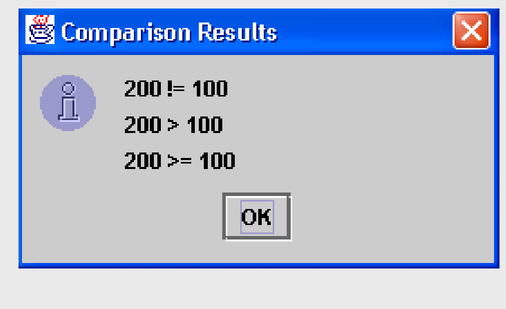
Input
Enter first integer:
100
OK Cancel

Input
Enter first integer:
200
OK Cancel

Comparison Results
100 != 200
100 < 200
100 <= 200
OK







شكل (2.20) مثال على عمليات المقارنة

شرح البرنامج

من خلال البرنامج يتضح أن فصل التطبيق Comparison بدأ تعريفه في السطر 8

```
public class Comparison {
```

كما قلنا سابقا فإن الطريقة main و التي كتبت من السطر 11 إلى السطر 59 يبدأ تنفيذها أولا في كل تطبيقات الجافا.

السطور من 13 إلى 17

```
String firstNumber;
String secondNumber;
String result;
int number1;
int number2;
```

هي عبارة عن تعريف للمتغيرات المستخدمة في الطريقة main

لاحظ أن هناك ٣ متغيرات من النوع String

firstNumber وهو الرقم الأول المدخل من المستخدم في صندوق الحوار ويدخل على شكل نص

secondNumber هو الرقم الثاني المدخل من المستخدم في صندوق الحوار ويدخل على شكل نص

result وهو متغير يخزن فيه الناتج على صورة نص

هناك أيضا ٢ متغير من النوع int

number1 وهو الرقم الأول الذي سيتم مقارنته

number2 وهو الرقم الثاني الذي سيتم مقارنته

لاحظ أن المتغيرات ذات النوع الواحد يمكن تعريفها في نفس السطر

مثال :

```
String firstNumber , secondNumber , result;
```

السطور 21-20

```
firstNumber =
JOptionPane.showInputDialog( "Enter first integer:" );
```

هذا الأمر يسمح للمستخدم بإدخال رقم داخل صندوق الحوار ويخزن داخل المتغير firstNumber على شكل نص string

السطور 24-25

```
secondNumber =
JOptionPane.showInputDialog( "Enter second integer:" );
```

هذا الأمر يسمح للمستخدم بإدخال الرقم الثاني من خلال صندوق حوار ويخزن في المتغير secondNumber على شكل نص String

السطور 28-29

```
number1 = Integer.parseInt( firstNumber );
number2 = Integer.parseInt( secondNumber );
```

كل من السطرين السابقين يقوم بتحويل قيمة نص String إلى قيمة صحيحة int ففي السطر 28 قمنا بتحويل قيمة المتغير firstNumber النصية إلى قيمة صحيحة ويخزنها في المتغير number1

أيضا السطر 29 حولت قيمة المتغير secondNumber النصية إلى قيمة صحيحة وخزنت في المتغير number2 .

السطر رقم 33

```
result = "";
```

في هذا السطر قمنا بتخصيص نص فارغ empty string إلى المتغير result وذلك لأن كل متغير يعرف داخل طريقة method لابد أن يأخذ قيمة ابتدائية قبل أن يستخدم لذلك تم تخصيص هذا النص الفارغ مؤقتا كقيمة ابتدائية. وهنا يجب التنويه عن هذا الخطأ الشائع الحدوث :

خطأ شائع : عدم إعطاء المتغير المعرف داخل طريقة method قيمة ابتدائية وذلك قبل استخدامه **داخل الطريقة يعطى syntax error**

السطر 34-35

```
if ( number1 == number2 )
result = number1 + " == " + number2;
```

هذه هي جملة if وعادة تبدأ جملة if بكلمة if يتبعها شرط داخل أقواس ثم يأتي بعد ذلك جملة هي في الواقع من التركيب الأساسي ل if لذلك كان يمكن كتابة السطر ٣٥ مع السطر ٣٤ دون فصلهما ولكن تم الفصل للسهولة أثناء القراءة ، ولذلك نلاحظ عدم وجود الفاصلة المنقوطة (;) والتي تدل على نهاية الجملة في السطر 34 ، معنى ذلك أن الجملة لم تنتهِ عند الشرط .

نفرض أن number1، number2 متساويين إذن تنفذ الجملة التالية

```
result = result + number1 + "==" + number2;
```

وفي هذا السطر تم تخصيص

```
result + number1 + "==" + number
```

للمتغير result.

وهنا نلاحظ وجود قيمتين صحيحتين هما number1، number2 فكيف يتم إضافة قيمة صحيحة إلى أخرى نصية string ثم تخزين الناتج في متغير أيضا نص string

في حقيقة الأمر هذه العملية تسمى string concatenation

يتم تحويل قيم number1، number2 إلى قيم نصية ثم تضاف إلى القيمة وتخزن الناتج في المتغير result.

أخطاء شائعة :

- عند تبديل العملية (=) مكان (==) في شرط جملة if يعطى syntax error
- وضع فاصلة منقوطة بعد الشرط مثل ; if (number1 == number2)

يعطى خطأ منطقي أي خطأ يظهر أثناء تنفيذ البرنامج وذلك لأنه يعتبر أن جواب الشرط جملة خالية.

أسئلة وتمارين

١) حدد أياً من الجمل التالية صح وأيها خطأ مع التعليل

ا - عند تنفيذ برنامج ما ، إذا وجدت ملاحظات فإن الكمبيوتر يقوم بطباعة النص الذي يأتي بعد // على الشاشة. ( )

- -

- -

ب - العامل موديلاس ٪ يمكن استخدامها فقط مع معاملات صحيحة ( )

- -

- -

ج - العمليات الحسابية التالية تأتي كلها في نفس مستوى أولوية التنفيذ - , + ,%, / , * ( )

- -

- -

د - الطريقة Integer.parseInt تحول رقماً صحيحاً int إلى نص string ( )

- -

- -

٢) أكتب جمل الجافا التي تحقق كل من المهام التالية:

أ - حول قيمة نصية String إلى قيمة صحيحة integer ثم خزن القيمة المحولة في متغير صحيح age ، وذلك بفرض أن النص مخزن في value .

- -
- -
- -

ب - إذا كان المتغير number لا يساوي 10 اعرض في صندوق الرسالة التالية:

“The variable number is not equal to 10”

- -
- -
- -

٣) حدد ما هو الخطأ في الجمل التالية:

ا -

```
if ( c < 7 );
 JoptionPane.showMessageDialog (null, “c is less than 7”);
```

- -
- -
- -

ب –

```
  if ( c => 7 )
  JoptionPane.showMessageDialog (null,“c is equal or greater than
7”);
```

- -
- -

٤ - اكتب برنامجاً تطبيقياً بلغة الجافا يقوم بسؤال المستخدم أن يُدخل رقمين، ثم يقوم بعد ذلك بطباعة الرقم الأكبر متبوعاً بالنص التالي " is larger " وذلك داخل صندوق رسالة. وإذا كانا الرقمان متساويين يطبع الرسالة " these number are equal ".

٥ - اكتب برنامجاً تطبيقياً بلغة الجافا يقوم بقراءة ثلاثة أرقام صحيحة من المستخدم، ثم يقوم بعرض المجموع، المتوسط الحسابي، حاصل الضرب، الرقم الأصغر والرقم الأكبر وذلك داخل صندوق رسالة.

٦ -اكتب برنامجاً تطبيقياً بلغة الجافا يقوم بقراءة نصف قطر دائرة من المستخدم، ثم يقوم بعد ذلك بطباعة قطر الدائرة، المحيط والمساحة.

افرض أن الثابت الطبيعي $\pi = 3.14159$

- ملحوظة: يمكنك استخدام الثابت Math. PI وذلك للثابت الطبيعي وهذه القيمة تعتبر أدق من القيمة 3.14159

٧ - اكتب برنامجاً تطبيقياً بلغة الجافا يقوم بقراءة 5 أرقام صحيحة من المستخدم، ثم يقوم بطباعة الرقم الأكبر والرقم الأصغر.

- ملحوظة: استخدم التقنيات التي تعلمتها في هذا الفصل فقط.

٨ - اكتب برنامجاً تطبيقياً بلغة الجافا يقوم بقراءة عدد صحيح من المستخدم، ثم يقوم بطباعة رسالة يحدد فيها ما إذا كان هذا العدد فردياً أم زوجياً.

- ملحوظة: استخدم عامل الموديلاس، أي عدد زوجي هو مضاعفات الرقم 2 لذلك أي مضاعف للعدد 2 يعطي باقياً 0 عند القسمة على 2.

٩ - اكتب برنامجاً تطبيقياً بلغة الجافا يقوم بقراءة عددين صحيحين من المستخدم، ثم يحدد ويطبع ما إذا كان الأول هو مضاعف الثاني.

١٠ - اكتب برنامجاً تطبيقياً بلغة الجافا يقوم بقراءة عدد من المستخدم مكون من 5 أرقام صحيحة، ثم يقوم البرنامج بعد ذلك بطباعة الأرقام وبين كل رقم وآخر مسافة.

مثال: إذا كان العدد هو 42339 يقوم البرنامج بطباعة 4 2 3 3 9

المملكة العربية السعودية
المؤسسة العامة للتعليم الفني والتدريب المهني
الإدارة العامة لتصميم وتطوير المناهج



برمجه الحاسب

أدوات التحكم البنائي

أدوات التحكم البنائي



الجدارة:

أن يكون المتدرب قادرا على كتابة الشفرة البرمجية code للبرامج المتوسطة نسبيا

الأهداف :

عندما تكمل هذه الوحدة يكون لديك القدرة على:

١. فهم ومعرفة جمل التفريع
٢. كتابة جمل if ، if/else ، if المتداخلة
٣. كتابة جملة switch
٤. معرفة الحلقات التكرارية
٥. كتابة واستخدام حلقة while
٦. كتابة واستخدام حلقة do/while
٧. كتابة واستخدام حلقة for
٨. كتابة واستخدام الحلقات المتداخلة
٩. كتابة برامج متوسطة

مستوي الأداء المطلوب:

أن يصل المتدرب إلى إتقان هذه الجدارة بنسبة 100 ٪

الوقت المتوقع للتدريب : ١٦ ساعة

الوسائل المساعدة :

- حاسب إلى
- قلم
- دفتر

متطلبات الجدارة :

اجتياز جميع الحقائب السابقة

أدوات التحكم البنائي

عادة يتم تنفيذ الجمل في البرامج بطريقة تتابعيه جملة بعد أخرى ولكن عند استخدام أداة من أدوات التحكم مثل if لا تنفذ الجملة إلا إذا تحقق الشرط السابق.

- في الفصل السابق درسنا شكلاً من أشكال اتخاذ القرار عن طريق استخدام جملة if في أبسط صورها وقلنا

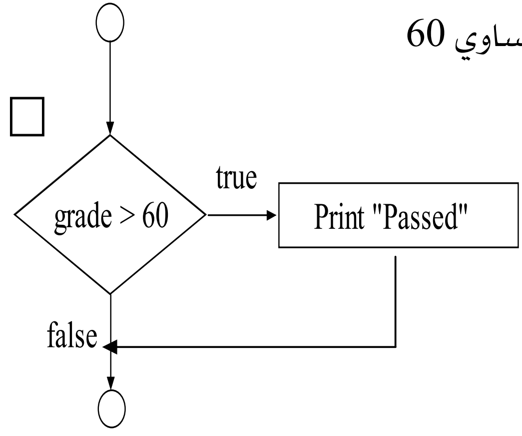
```
If (number1 == number2 )
result = result+ “ number1 == number2 “ ;
```

جملة if السابقة تسمى if selection structure حيث إنها تقوم بانتخاب الجملة التي يتم تنفيذها

فمثلا : افرض أن درجة اجتياز أي اختبار هي 60 من 100 تكتب جملة if على النحو

```
If (studentGrade > = 60 )
System.out.println (“ passed” );
```

وبذلك نرى أن كلمة (“ passed “) لن يتم طباعتها على الشاشة إلا إذا كانت درجة الطالب في الاختبار أكبر من أو تساوي 60

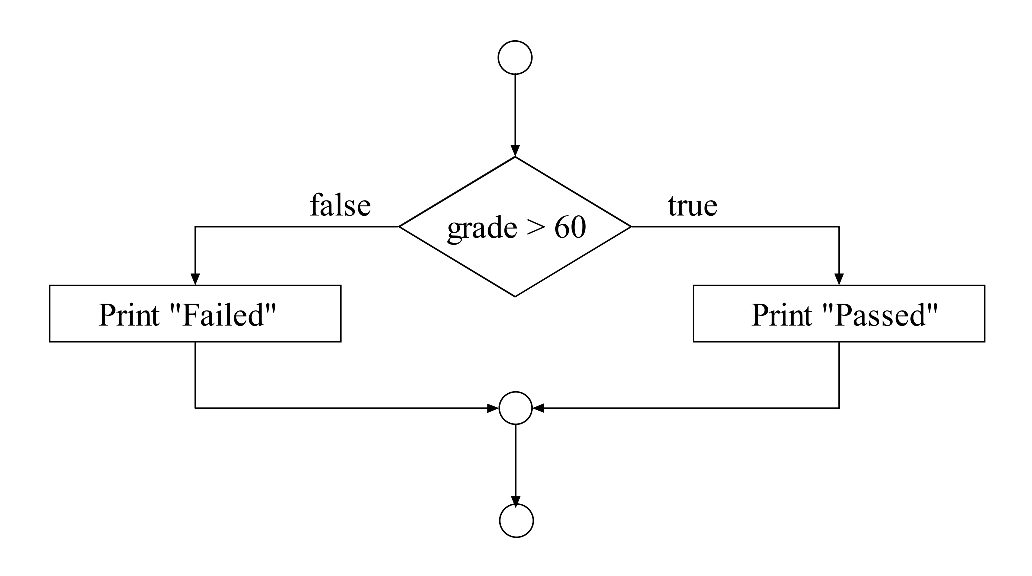


شكل (1-3) خريطة تدفق لجملة if

من الشكل السابق يتضح أن

- إذا تحقق الشرط نفذت الجملة طباعة كلمة (“ passed “) ثم تنفيذ باقي جمل البرنامج
- إذا لم يتحقق الشرط لم تنفذ الجملة وأكمل تنفيذ باقي جمل البرنامج.

جملة if \ else



شكل (2-3) خريطة تدفق لجملة if/else

يمكننا استخدام جملة if \ else إذا أردنا

- عند تحقق الشرط تنفذ جملة ما ثم ينفذ باقي البرنامج
- عند عدم تحقق الشرط تنفذ جملة أخرى ثم بعدها ينفذ باقي البرنامج

مثال

```
if ( studentGrade > = 60 )
System.out.println ( " passed " );
else
System.out.println ( " failed " );
```

من هذا المثال يتضح أنه إذا كان تقدير الطالب أكبر من أو يساوي 60 طبعت على الشاشة كلمة " passed" أما إذا كانت الدرجة غير ذلك أي أقل من 60 طبعت كلمة " failed "

العملية (? :)

يمكن في لغة الجافا اختصار جملة if \ else بالعملية (?:) وبذلك يمكن التعبير عن جملة if \ else السابقة كالتالي

```
System.out.println (studentGrade >= 60 ? " passed " : " failed " ) ;
```

لاحظ أن **?** تمثل جملة if ، **:** تمثل else

جمل if \ else المتعددة

يمكن استخدام جمل if \ else المتعددة وذلك في حالات الاختيار من متعدد

مثال : المطلوب طباعة a إذا كانت درجة الاختبار أكبر من أو تساوي 90 ، b إذا كانت الدرجة أكبر من أو تساوي 80 ، c إذا كانت الدرجة من 70 إلى 79 ، d إذا كانت الدرجة من 69 إلى 60 ويطبع f غير ذلك :

```
if (studentGrade >= 90 )
System.out.println ( " a " );
else if ( studentGrade >= 80 )
System.out.println ( " b " );
else if ( studentGrade >= 70 )
System.out.println ( " c " );
else if ( studentGrade >= 60 )
System.out.println ( " d " );
else
System.out.println ( " f " );
```

لاحظ أنه عند تحقق الشرط في جملة من الجمل السابقة تنفذ الجملة التالية فقط .

أما إذا أردنا تنفيذ أكثر من جملة لابد من عمل أقواس { }

فمثلا إذا أردنا في المثال السابق أن يطبع f وجملة " you must take this course again "

فتكون الجملة الأخيرة على الشكل

```
else
    {
        System.out.println ( " f " );
        System.out.println ( " you must take this course again " );
    }
```

لاحظ أنه عند وجود الأقواس يتم التعامل مع ما بداخلها على أنها جملة واحدة

خطأ شائع :

عند نسيان أحد الأقواس أو كليهما يؤدي إلى وجود خطأ في بناء الجملة syntax error أو خطأ منطقي ففي المثال السابق عند نسيان أحد الأقواس يعطي syntax error أما عند نسيان القوسين معا فإنه لا يعطي خطأ عند عمل ترجمة compile ولكن يعطي خطأ منطقياً أي عند التنفيذ إذا كانت الدرجة أقل من 60 يطبع f ولكن لا يطبع الجملة " you must take this course "

عند وضع فاصلة منقوطة (;) بعد الشرط يعطي خطأ منطقي logical error إذا كنا نستخدم جملة if البسيطة ، أما إذا كنا نستخدم جمل if المتعددة يعطي syntax error

استخدام جملة switch

في حالات الاختيار من متعدد يمكن أن تحل جملة switch محل if/else المتعددة

تكتب جملة switch على الشكل التالي:

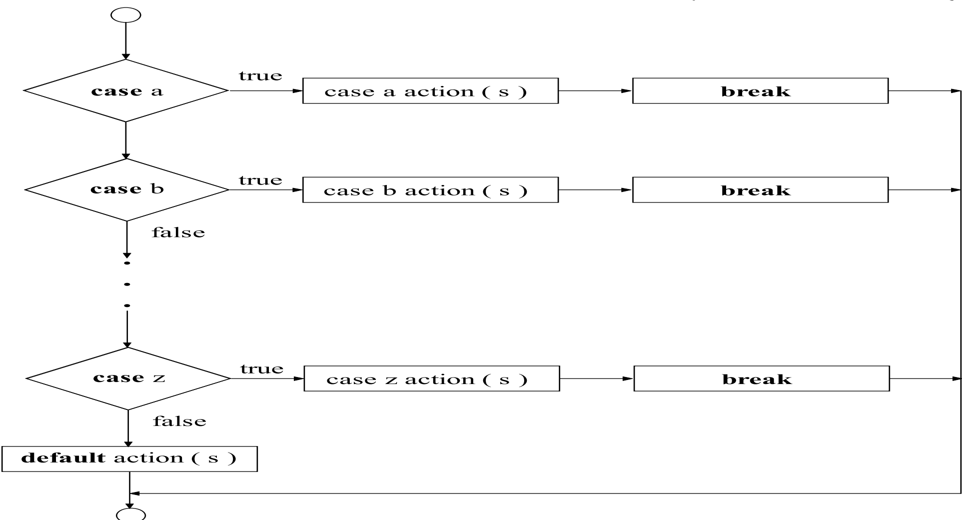
```
switch (switch-expression)
{ case value1: statement(s)1;break;
  case value2: statement(s)2;break;
  …
  case valueN: statement(s)N;break;
        default :          statement(s)- for – default;
}
```

لاحظ أن switch expression لابد أن تكون قيمة (charأو byte أو short أوint)

لاحظ أيضا أن قيم value1,value2,…,valueN لابد أن تكون من نفس نوع switch expression

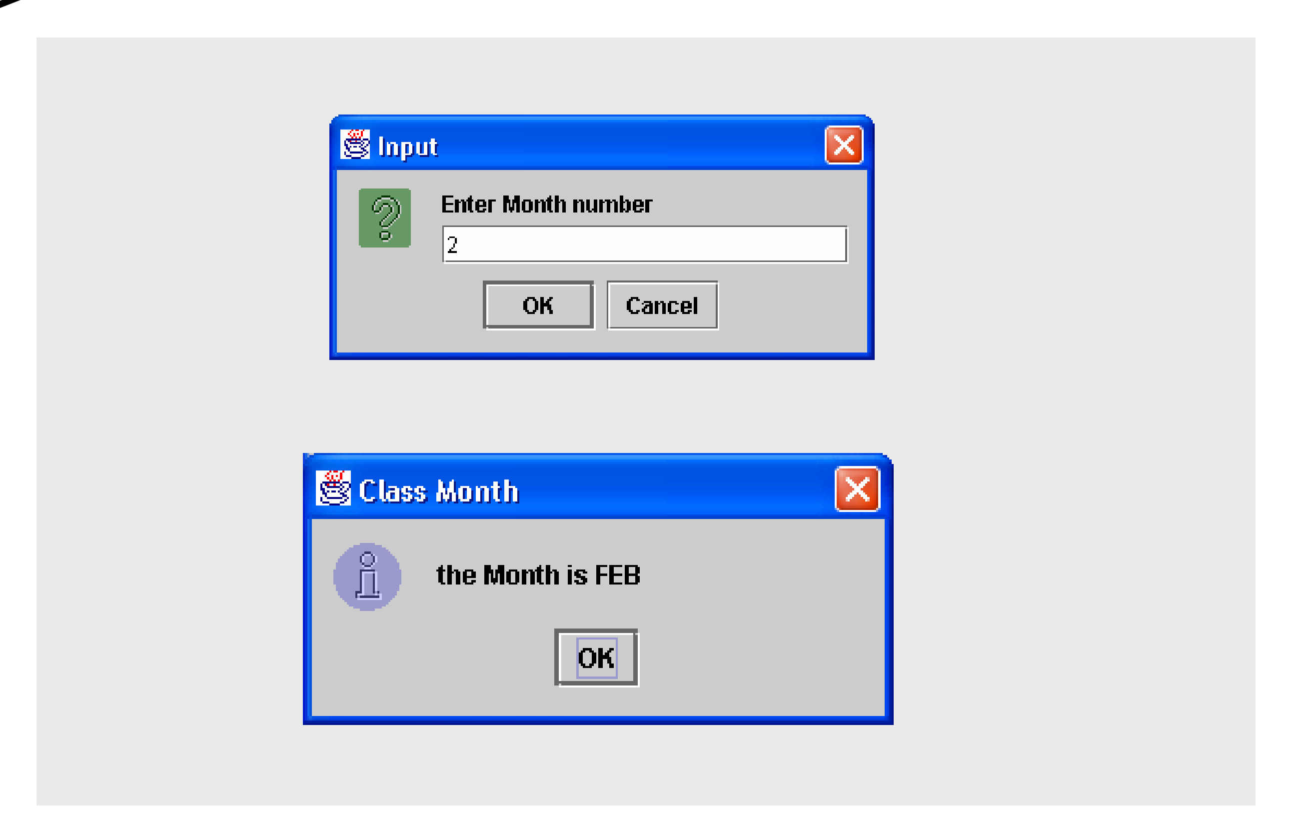
الجملة(الجمل) statement(s) تنفذ إذا وإذا فقط كانت القيمة value تساوي switch expression ، ثم تأتي بعد ذلك جملة break لكي تعمل هروب من جملة switch

الحالة الافتراضية default وهي اختيارية ويمكن أن تستخدم لتنفيذ مهمة في حالة عدم تحقق أي حالة ودائما تكون هذه الجملة في آخر جملة switch .



شكل (3-3) خريطة تدفق جملة switch

```
1.  // Fig. 3.4 : Month.java
2.  // Class Month program with switch statements.
3.  // Java extension packages
4.  import javax.swing.JOptionPane;
5.
6.  public class Month {
7.  // main method begins execution of Java application
8.  public static void main( String args[] )
9.  {
10. int month;  // number of month number entered
11. String input;      // month number typed by user
12. String name;        // name of month
13.
14. // Processing phase
15. // prompt for input and read Month number from user
16. input = JOptionPane.showInputDialog(
17. "Enter Month number" );
18. // convert grade from a String to an integer
19. month = Integer.parseInt( input );
20. switch ( month )
21. {
22. case 1:name="JAN";break;
23. case 2:name="FEB";break;
24. case 3:name="MAR";break;
25. case 4:name="APR";break;
26. case 5:name="MAY";break;
27. case 6:name="JUN";break;
28. case 7:name="JUL";break;
29. case 8:name="AUG";break;
30. case 9:name="SEP";break;
31. case 10:name="OCT";break;
32. case 11:name="NOV";break;
33. case 12:name="DEC";break;
34.
35. default :name=" invalid Month number ";
36. }
37.
38. // display name of month number
39. JOptionPane.showMessageDialog( null,
40. "the Month is " + name ,
41. "Class Month", JOptionPane.INFORMATION_MESSAGE );
42.
43. System.exit( 0 );    // terminate application
44.
45. }  // end method main
46.
```



شكل (3-4) مثال على جملة switch

شرح البرنامج

البرنامج يقوم بقراءة رقم من المستخدم ثم يقوم بطباعة اسم الشهر المقابل له لاحظ وجود جملة switch في السطور من ٢٠ إلى ٣٦ في حالة تطابق الرقم المدخل مع أي حالة من حالات جملة switch يتم تنفيذ الجملة التي تلي الحالة فمثلا إذا كان الرقم المدخل هو ٢ فهذا الرقم يتطابق مع الحالة case 2: لذلك يتم تنفيذ الجملة التالية وهي;name="FEB" وهي تخصيص السلسلة FEB إلى المتغير name .

لاحظ وجود جملة ;break والتي تتسبب في الهروب من جملة switch .

في حالة عدم تطابق الرقم المدخل من المستخدم مع الحالات الموجودة يتم تنفيذ الحالة الافتراضية الموجودة في السطر رقم 35

default :name=" invalid Month number ";

تخصيص النص السابق للمتغير name .

السطر ٣٦ يحتوي على } وهي تمثل نهاية جملة switch.

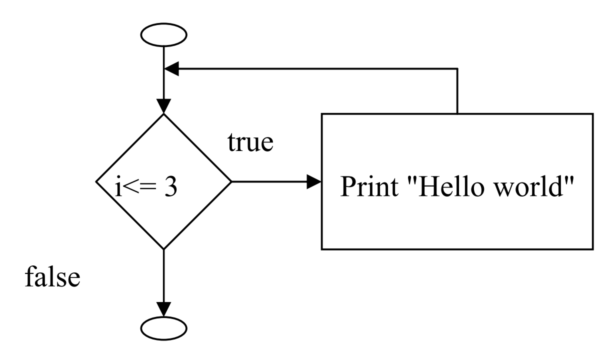
السطور من ٣٩ إلى ٤١

```
JOptionPane.showMessageDialog( null,"the Month is " + name ,
"Class Month", JOptionPane.INFORMATION_MESSAGE );
```

جملة طباعة للمتغير في صندوق رسالة

بناء حلقة **while** التكرارية

تسمح الجمل التكرارية للمبرمج أن يعرف جملة ما أو عدة جمل أن يحدث لها تكرار طالما أن الشرط صحيح



شكل (3-5) خريطة تدفق حلقة while

مثال إذا أردنا طباعة جملة " hello world " ٣ مرات

```
int   i= 1 ;
While   ( i < = 3 )
{
System.out.println  ( " hello world " ) ;
i + = 1 ;
}
```

نجد من المثال السابق أن جملة التكرار while لها ٣ بنود

١ - جملة الإشعال : وهي تعريف المتغير و إعطاؤه قيمة ابتدائية هي ١

٢ - الشرط : وهو شرط حدوث التكرار وهو أن تكون i أقل من أو تساوي ٣

٣ - زيادة العداد : وفي المثال السابق تتم زيادة i بمقدار ١ بعد جملة الطباعة

لاحظ وجود الأقواس وذلك لأننا نريد تكرار أكثر من جملة وهي جملة الطباعة وجملة زيادة العداد.

مثال

```
int sum = 0 ;
int i = 1
While ( i < = 10 )
{       sum + = i ;
        i + = 1 ;
}
System.out.println ( sum )
```

وفي المثال السابق أردنا طباعة مجموعة الأعداد من ١ إلى ١٠

لاحظ وجود جملة الطباعة خارج الأقواس .

خطأ شائع : عدم وضع جملة تجعل الشرط لجملة while خطأ يعتبر خطأ منطقياً لأنه يجعل الحلقة تستمر إلى ما لانهاية وذلك لأن الشرط يبقى دائما صحيحاً.

فمثلا في المثال السابق عند إلغاء جملة زيادة العداد تظل دائما قيمة المتغير i هي ١ ويستمر الشرط صحيحاً وتستمر الحلقة إلى ما لانهاية.

- أيضا كتابة while بحرف كبير مثل While يعتبر خطأ في بناء الجملة ، وتذكر أن لغة الجافا تفرق بين الحروف الكبيرة والصغيرة وأن جميع الكلمات المحجوزة reserved keywords مثل if – else – while - ……… تحتوي كلها على حروف صغيرة .

مثال

```
1. // Fig. 3.6: Average1.java
2. // Class average program with counter-controlled repetition.
3. // Java extension packages
4. import javax.swing.JOptionPane;
5.
6. public class Average1 {
7. // main method begins execution of Java application
8. public static void main( String args[] )
9. {
10. int total,          // sum of grades input by user
11. gradeCounter,   // number of grades entered
12. gradeValue,     // grade value
13. average;        // average of all grades
14. String grade;       // grade typed by user
15.
16. // Initialization Phase
17. total = 0;          // clear total
18. gradeCounter = 1;   // prepare to loop
19.
20. // Processing Phase
21. while ( gradeCounter <= 10 ) {  // loop 10 times
22.
23. // prompt for input and read grade from user
24. grade = JOptionPane.showInputDialog(
25. "Enter integer grade: " );
26. // convert grade from a String to an integer
27. gradeValue = Integer.parseInt( grade );
28.
29. // add gradeValue to total
30. total = total + gradeValue;
31.
32. // add 1 to gradeCounter
33. gradeCounter = gradeCounter + 1;
34.
35. }  // end while structure
36.
37. // Termination Phase
38. average = total / 10;  // perform integer division
39.
40. // display average of exam grades
41. JOptionPane.showMessageDialog( null,
42. "Class average is " + average, "Class Average",
43. JOptionPane.INFORMATION_MESSAGE );
44. System.exit( 0 );     // terminate the program
45. }  // end method main
46. }  // end class Average1
```

Input
Enter integer grade:
100
OK Cancel

Input
Enter integer grade:
88
OK Cancel

Input
Enter integer grade:
93
OK Cancel

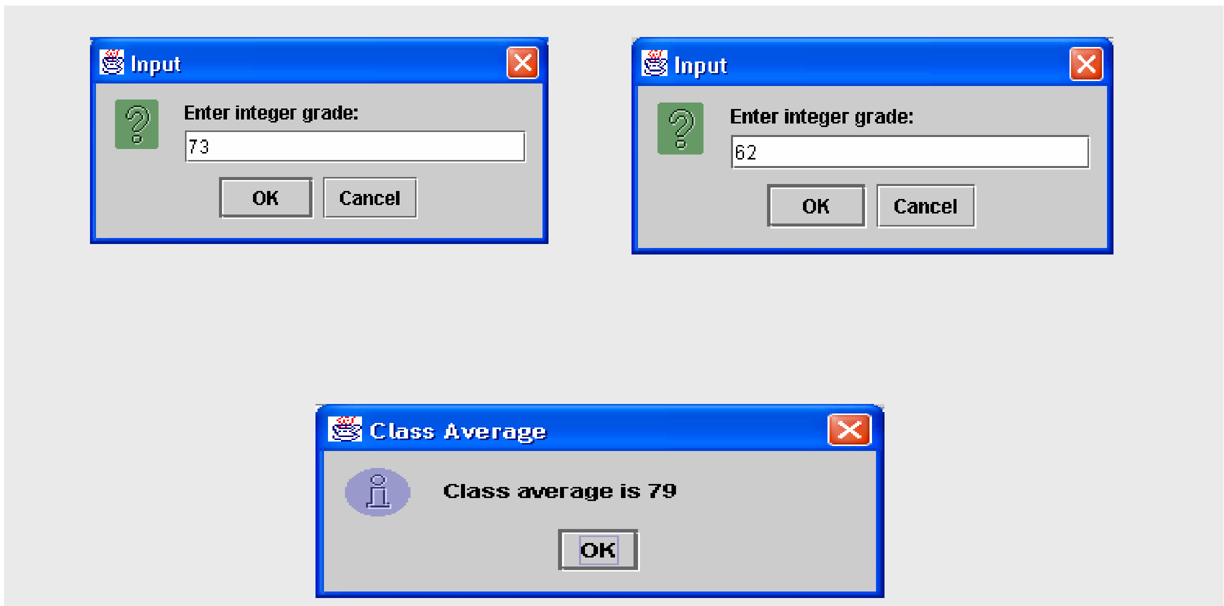
Input
Enter integer grade:
55
OK Cancel

Input
Enter integer grade:
68
OK Cancel

Input
Enter integer grade:
77
OK Cancel

Input
Enter integer grade:
83
OK Cancel

Input
Enter integer grade:
95
OK Cancel



شكل (3-6) مثال عل استخدام حلقة while

شرح البرنامج

المثال السابق يقوم بأخذ تقديرات الطلاب في مادة ما ويقوم بطباعة المتوسط الحسابي للتقديرات.

السطر رقم ٤ import javax.swing.JOptionPane ;

تقوم هذه الجملة بعمل استيراد للفصل JOptionPane وذلك لكي يسمح للبرنامج بقراءة البيانات من لوحة المفاتيح ، أيضا يسمح بعرض ناتج التنفيذ على الشاشة وذلك من خلال صندوق حوار السطر رقم 6 public class Averagel {

تعريف اسم الفصل Averagel تذكر أن أي تطبيق لابد أن يحتوى على الطريقة main لكي يتم تنفيذ التطبيق الطريقة main في هذا المثال من (السطر 8 إلى 45) السطور من 10 إلى 14 تقوم بتعريف المتغيرات

total , gradeCounter , gradeValue , average

من النوع الصحيح integer

أيضا تقوم بتعريف المتغير grade من نوع متسلسلة String والذي يقوم بحفظ النص الذي يدخله المستخدم في صندوق الحوار

المتغير gradeValue يحفظ القيمة الصحيحة للنص المخزن في المتغير grade وذلك بعد عملية التحويل التي سيجريها البرنامج على النص.

لاحظ أن كل المتغيرات السابقة تم تعريفها داخل الطريقة main وهي متغيرات محلية أي تستخدم فقط داخل هذه الطريقةmain ولا يمكن التعامل معها من طريقة أخرى .

أيضا لا يجب استخدام متغير ما قبل تعريفه.

السطور من 17 , 18

```
total = 0;          // clear total
gradeCounter = 1;   // prepare to loop
```

هي جمل تخصيص تقوم بإشغال المتغيرين gradeCounter , total وإعطائهما قيمتين ابتدائيتين 0، 1 على الترتيب . وذلك قبل استخدامها في حسابات البرنامج

السطر 21

```
while ( gradeCounter <= 10 ) {
```

بداية حلقة while التكرارية وتحدد شرط استمرار الحلقة وهو أن تكون gradeCounter أقل من أو تساوي 10 .

لاحظ وجود القوس } وهو يمثل بداية الجمل المراد تكرارها داخل الحلقة .

خطأ شائع : استخدام المتغير في الحسابات قبل إعطائه قيمة ابتدائية يظهر رسالة الخطأ التالية

variable may not have been initialized

وهو خطأ يوضح أن المتغير يجب إشعاله وإعطاؤه قيمة ابتدائية قبل استخدامه ، ففي المثال السابق إذا أهملنا السطر رقم ١٨ الخاص بإشعال المتغير gradeCounter ، ثم نستخدم نفس المتغير في السطر ٢١ في الشرط لا يمكن للبرنامج من تحديد قيمة gradeCounter وبالتالي يعطي خطأ السطور 24، 25

```
grade = JOptionPane.showInputDialog("Enter integer grade: " );
```

تقوم بعرض صندوق حوار للمستخدم يطلب من إدخال تقدير الطالب ،تخزن القيمة المدخلة في المتغير grade يتم تحويل النص إلى عدد صحيح ويحفظ في المتغير gradeValue عن طريق السطر 27

```
gradeValue = Integer . parseInt ( grade ) ;
```

تذكر أن الفصل Integer يوجد داخل الحزمة java.lang والتي يقوم المترجم باستيرادها أوتوماتيكيا مع كل برنامج جافا دون الحاجة لكتابة جملة استيراد import في بداية البرنامج .
السطر 30

```
total = total + gradeValue ;
```

يقوم بعملية جمع الدرجات وتخزينها في المتغير total
السطر33

```
gradeCounter = gradeCounter + 1;
```

زيادة العداد بمقدار ١ السطر 35
نهاية جملة

```
while }
```

يتم تكرار السطور من 21 إلى 35 حتى تصبح قيمة gradeCounter أكبر من 10 عند ذلك نقف الحلقة التكرارية وينتقل البرنامج لتنفيذ السطر التالي السطر 38

```
average = total / 10;
```

يقوم هذا السطر بحساب قيمة المتوسط الحسابي وذلك بقسمة مجموع الدرجات total على عددها 10 .

السطور 41 ، 42،

```
43 JOptionPane.showMessageDialog( null, "Class average is " + average,
"Class Average", JOptionPane.INFORMATION_MESSAGE );
```

تمثل جملة الطباعة بواسطة صندوق الحوار .

لاحظ أنه في المثال السابق كان عدد مرات التكرار محدودة ومعروفة مسبقا = 10 وذلك عن طريق الشرط

```
while ( gradeCounter < = 10 )
```

وبذلك حددنا لمستخدم البرنامج أن يقوم بإدخال 10 أرقام .

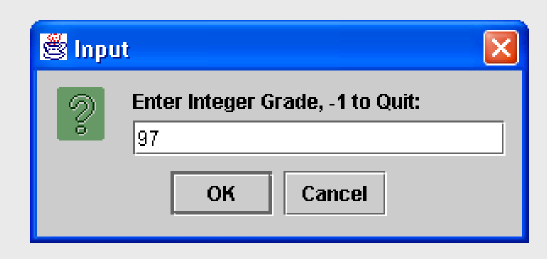
في المثال القادم سوف ترى أنه من الممكن أن تستمر الحلقة التكرارية ويستمر البرنامج في سؤال المستخدم أن يدخل درجة الطالب حتى يقوم المستخدم بإدخال رقم ١ وهو شرط توقف الحلقة

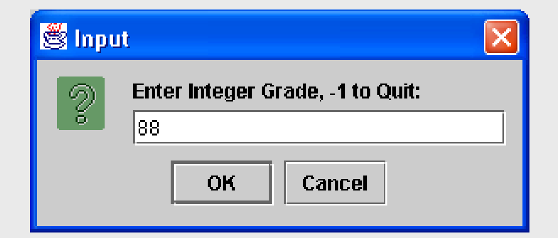
```
while  ( gradeCounter ! = - 1 )
{
//الجمل المراد تكرارها
   }
```

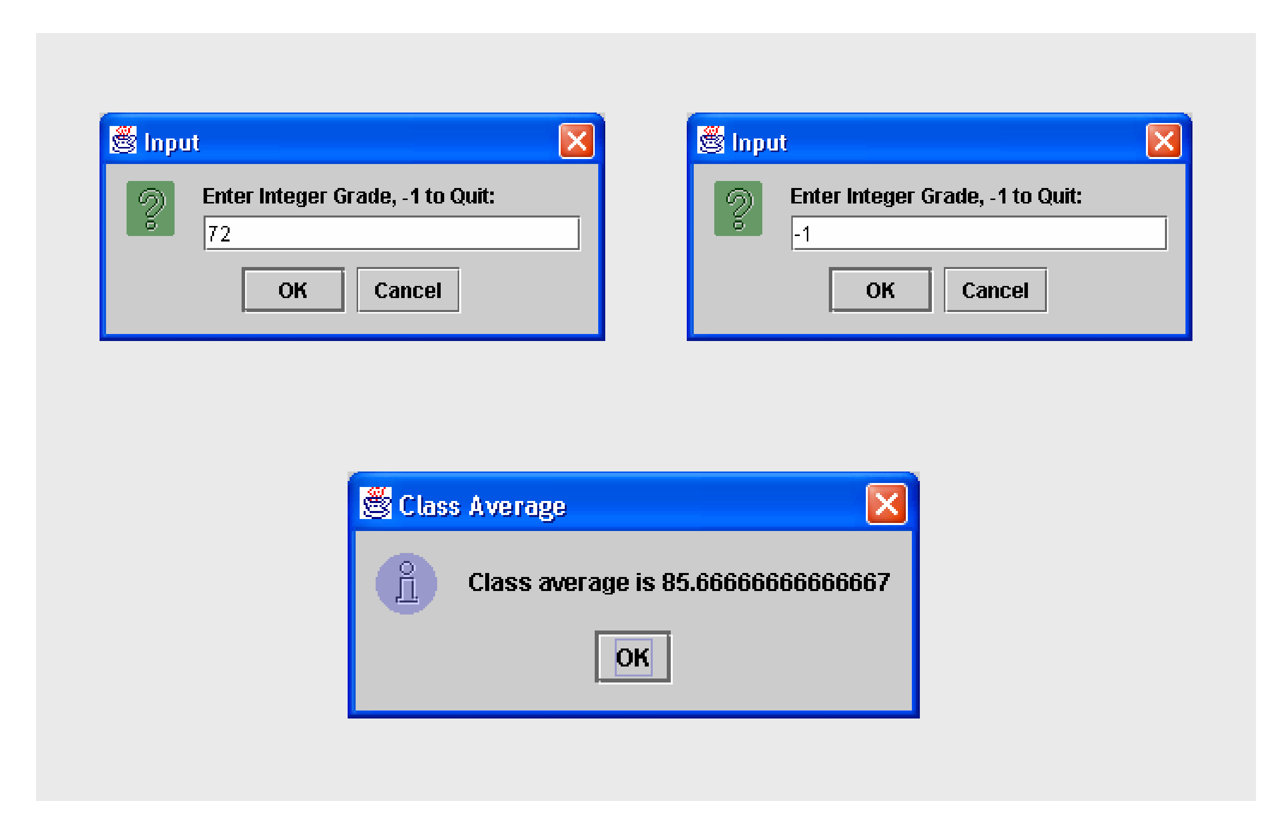
مثال

```
1.  // Fig. 3.7: Average2.java
2.  // Class average program with sentinel-controlled repetition.
3.  // Java extension packages
4.  import javax.swing.JOptionPane;
5.  public class Average2 {
6.  // main method begins execution of Java application
7.  public static void main( String args[] )
8.  {
9.  int gradeCounter,  // number of grades entered
10. gradeValue,    // grade value
11. total;         // sum of grades
12. double average;    // average of all grades
13. String input;      // grade typed by user
```

```
14. // Initialization phase
15. total = 0;          // clear total
16. gradeCounter = 0;  // prepare to loop
17. // Processing phase
18. // prompt for input and read grade from user
19. input = JOptionPane.showInputDialog(
20. "Enter Integer Grade, -1 to Quit:" );
21. gradeValue = Integer.parseInt( input );
22. while ( gradeValue != -1 ) {
23. total = total + gradeValue;
24. gradeCounter = gradeCounter + 1;
25.
26. // prompt for input and read grade from user
27. input = JOptionPane.showInputDialog(
28. "Enter Integer Grade, -1 to Quit:" );
29.
30. // convert grade from a String to an integer
31. gradeValue = Integer.parseInt( input );
32. }
33. if ( gradeCounter != 0 ) {
34. average = (double) total / gradeCounter;
35.
36. // display average of exam grades
37. JOptionPane.showMessageDialog( null,
38. "Class average is " + average,
39. "Class Average", JOptionPane.INFORMATION_MESSAGE );
40. }
41. else
42. JOptionPane.showMessageDialog( null,
43. "No grades were entered", "Class Average",
44. JOptionPane.INFORMATION_MESSAGE );
45. System.exit( 0 );    // terminate application
46. }  // end method main
47. }  // end class Average2
```







شكل (3-7) مثال على استخدام حلقة while

شرح البرنامج

السطور 19 -21

```
input = JOptionPane.showInputDialog(
"Enter Integer Grade, -1 to Quit:" );
gradeValue = Integer.parseInt( input );
```

البرنامج يقرأ قيمة من المستخدم ويقوم بتحويلها إلى القيمة الصحيحة يحتوي البرنامج على حلقة while التكرارية على الشكل

```
 while (gradeValue ! = -1 )
{
 // الجمل المراد تكرارها
}
```

تتكرر الجمل داخل الحلقة طالما شرط الاستمرار متحقق وهو أن تكون الدرجة لا تساوي ١-

23 السطر gradeValue ! = -1

total = total + gradeValue ; إذا كان الشرط متحققاً يقوم البرنامج بإضافة الدرجة إلى المتغير total

السطر 24

gradeValue = gradeValue + 1 ; يقوم بزيادة العداد بمقدار الوحدة وذلك لمعرفة عدد الدرجات السطور 27 - 31

```
input = JOptionPane.showInputDialog(
"Enter Integer Grade, -1 to Quit:" );

// convert grade from a String to an integer
gradeValue = Integer.parseInt( input );
```

يتم قراءة درجة أخرى من المستخدم ثم يتم تحويلها إلى القيمة الصحيحة int .

السطر 33–40

```
if ( gradeCounter != 0 ) {
average = (double) total / gradeCounter;
JOptionPane.showMessageDialog( null,
"Class average is " + average,
"Class Average", JOptionPane.INFORMATION_MESSAGE );
}
```

إذا كان المتغير gradeCounter لا يساوى صفراً ومعنى ذلك أن هناك درجات تم قراؤتها أو أن هناك درجة واحدة على الأقل تم قراؤتها ، إذا تحقق هذا الشرط يتم تنفيذ الجملة في السطر التالي وهو ٣٤ وهى لحساب المتوسط الحسابي ، لاحظ أننا استخدمنا الأمر (double) وذلك لأن كلاً من المتغير gradeCounter , total معرفين على أنهما صحيحان int وبما أن المتوسط الحسابي يمكن أن يكون قيمة ذات كسور عشرية لذلك استخدمنا الأمر (double) .ثم بعد ذلك يتم طباعة قيمة المتوسط الحسابي في صندوق حوار .

السطور 42 - 44

```
JOptionPane.showMessageDialog (null ,” no grades were entered “
(“ class average “ , JOptionPane.INFORMATION_MESSAGE ) ;
```

هي جملة طباعة يتم تنفيذها في حالة عدم إدخال أي درجة

السطر 45

```
System.exit ( 0 ) ;
```

يتم إضافة هذه الجملة عند استعمال الفصل JOptionPane

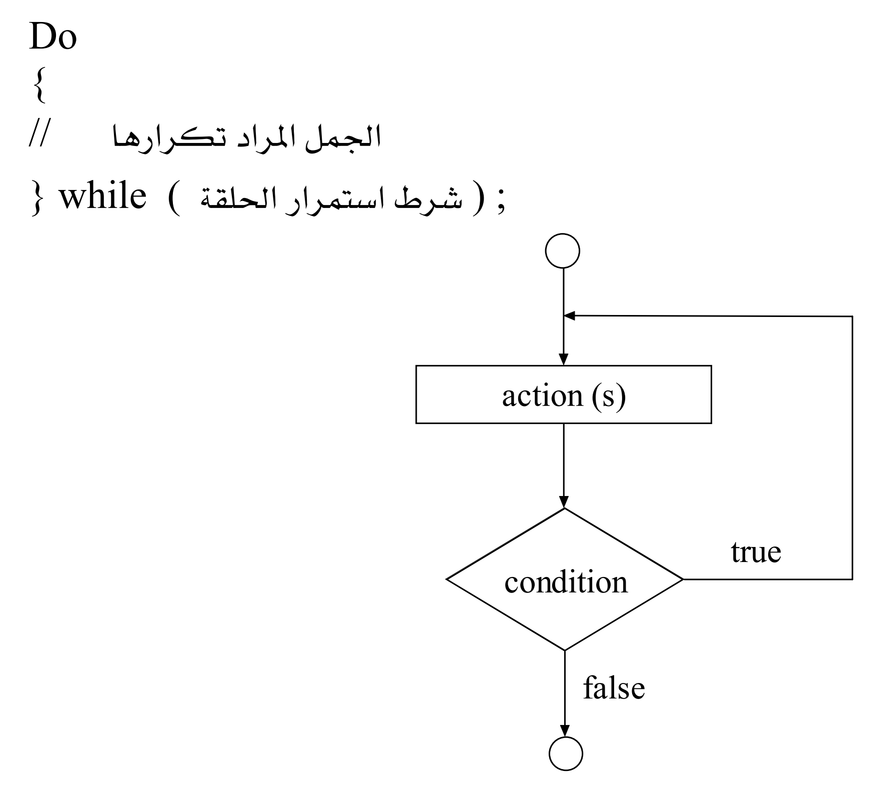
خطأ شائع:

عدم كتابة الأقواس بعد جملة while وذلك إذا كان المطلوب تكرار أكثر من جملة يعطى خطأ منطقي وذلك لأنه إذا لم يوجد أقواس يتم تكرار الجملة التالية لحلقة while مباشرة فقط .

حلقة do/while التكرارية

تستخدم حلقة do–while كسابقتها while لعمل تكرار لجملة أو عدة جمل ويكون التركيب البنائي لها على الشكل :

```
Do
{
//      الجمل المراد تكرارها
} while ( شرط استمرار الحلقة ) ;
```



شكل (3-8) خريطة تدفق حلقة do/while

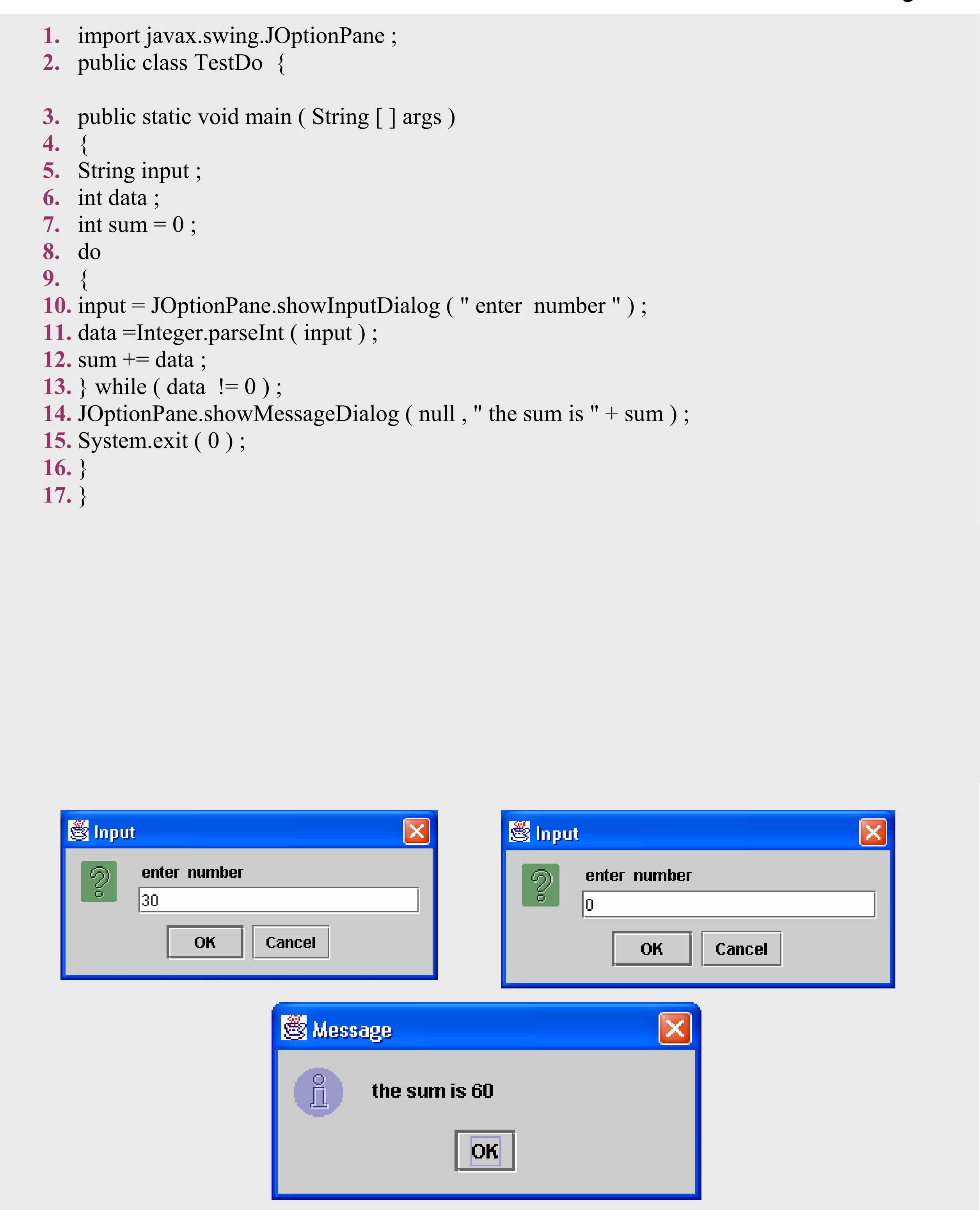
لاحظ أن الجمل المراد تكرارها تنفذ مرة واحدة على الأقل قبل أن يتم اختبار شرط استمرار الحلقة والذي يكون بداخل الأقواس بعد while فإذا كان الشرط صحيحاً true يتم التكرار والعودة لتنفيذ الجمل أما إذا كان خطأ false تتوقف الحلقة فورا . لذلك فإننا نرى الفرق بين جملة while وجملة do /while وهو أن الجمل يتم تنفيذها مرة واحدة على الأقل حتى لو كان شرط استمرار الحلقة خطأ false وذلك على عكس while التي تختبر الشرط أولا فإذا كان صحيحا يتم التنفيذ والتكرار وإذا كان خطأ تتوقف فورا دون تنفيذ الجمل داخل الحلقة .

المثال التالي يوضح فكرة عمل الحلقة do/while

مثال

```
1.  import javax.swing.JOptionPane ;
2.  public class TestDo  {

3.  public static void main ( String [ ] args )
4.  {
5.  String input ;
6.  int data ;
7.  int sum = 0 ;
8.  do
9.  {
10. input = JOptionPane.showInputDialog ( " enter  number " ) ;
11. data =Integer.parseInt ( input ) ;
12. sum += data ;
13. } while ( data  != 0 ) ;
14. JOptionPane.showMessageDialog ( null , " the sum is " + sum ) ;
15. System.exit ( 0 ) ;
16. }
17. }
```



شكل (3-9) مثال على استخدام حلقة do/while

هذا البرنامج يقوم بقراءة أرقام من المستخدم عن طريق توجيه رسالة له في صندوق حوار ثم إذا أدخل المستخدم رقم صفر يقوم البرنامج بطباعة حاصل جمع الأرقام المدخلة في صندوق رسالة .

لاحظ استخدام جملة do /while في السطور من ٨ إلى ١٣

داخل الحلقة تم توجيه رسالة للمستخدم ليدخل رقماً أو يدخل صفراً عند الانتهاء

السطر رقم ١١

```
data =Integer.parseInt ( input ) ;
```

تحويل القيمة من النوع سلسلة String إلى النوع الصحيح int

السطر رقم ١٢

```
sum += data ;
```

هو عبارة عن عملية الجمع ويتم إضافة الرقم الصحيح إلى المتغير sum

السطر رقم ١٣

```
} while ( data ! = 0 ) ;
```

هو عبارة عن إغلاق القوس } ثم يأتي بعد ذلك اختبار شرط استمرار الحلقة وهو هل قيمة الرقم المدخل لا تساوي صفراً فإذا كان الجواب بنعم يتم إعادة تكرار الحلقة وإن كان الجواب بلا فتتوقف الحلقة فورا وننتقل إلى السطر التالي.

السطر رقم ١٤

```
JOptionPane.showMessageDialog ( null , “ the sum is “ + sum ) ;
```

هو جملة طباعة في صندوق رسالة يتم فيها طباعة قيمة المتغير sum وهى حاصل جمع الأعداد المدخلة من قبل المستخدم

السطر رقم ١٥

```
System.exit ( 0 )
```

كما قلنا سابقا هذا الأمر يكتب في النهاية عند استخدام الفصل JoptionPane

ملحوظة : في هذا المثال أيضا كان عدد مرات تكرار الحلقة غير معلوم .

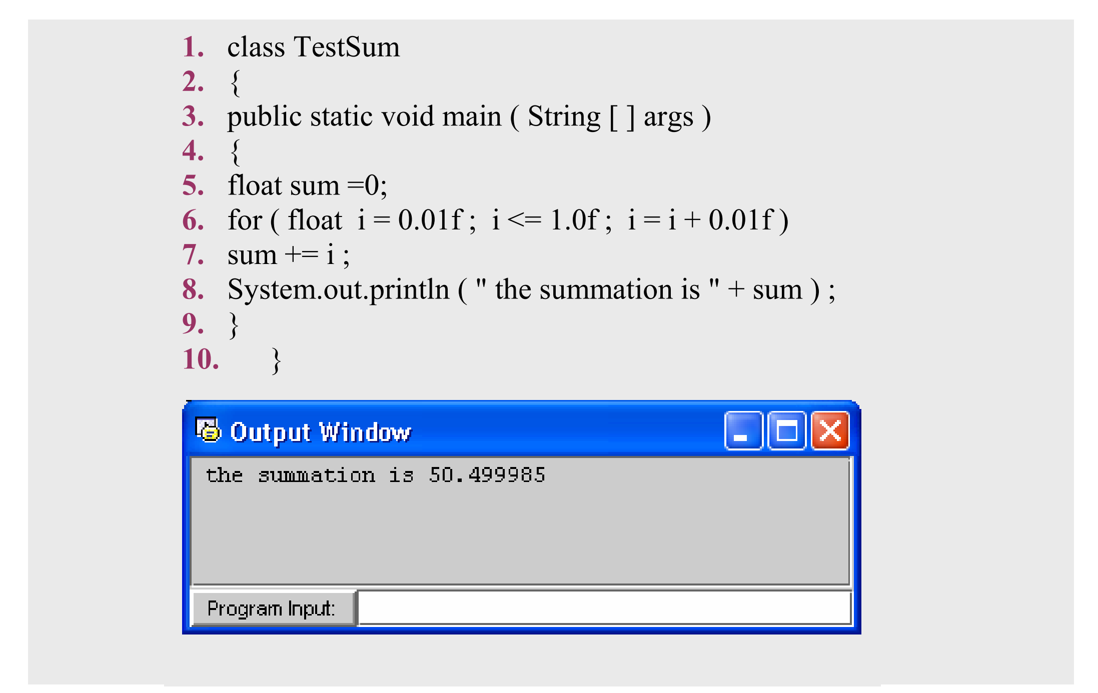
حلقة for التكرارية

تستخدم أيضا هذه الجملة لعمل تكرار لجملة أو عدة جمل ويكون التركيب البنائى لها على الشكل :

for (إعطاء قيمة ابتدائية للعداد ; شرط استمرار الحلقة ; جملة زيادة أو نقصان العداد)
{
 الجملة المراد تكرارها//
}

i = 0.01f
i <= 1.0f
true
sum = sum + i
i = i+1
false

شكل (3-10) خريطة تدفق حلقة for



```
1.  class TestSum
2.  {
3.  public static void main ( String [ ] args )
4.  {
5.  float sum =0;
6.  for ( float  i = 0.01f ;  i <= 1.0f ;  i = i + 0.01f )
7.  sum += i ;
8.  System.out.println ( " the summation is " + sum ) ;
9.  }
10.     }
```


شكل (3-11) مثال على استخدام حلقة for

شرح البرنامج

هذا البرنامج يقوم بجمع الأعداد العشرية من 0.01 وحتى 1.0

السطر رقم 5

```
float sum =0;
```

تعريف المتغير sum من النوع float أي يقبل الكسور العشرية و تم إعطاؤه قيمة ابتدائية صفر

السطر رقم 6

```
for ( float  i = 0.01f ;  i <= 1.0f ;  i = i + 0.01f )
```

جملة for التكرارية وتتكون من ٣ أجزاء :

الجزء الأول

```
float i = 0.01f
```

تعريف العداد وإعطاؤه قيمة ابتدائية تساوي 0.01

الجزء الثاني

```
i < = 1.0f
```

شرط استمرار الحلقة وهو أن يكون العداد i أقل من أو يساوي 1.0 بمعنى أنة إذا زادت قيمة العداد عن 1.0 تتوقف الحلقة فورا .

الجزء الثالث

```
i = i + 0.01f
```

زيادة العداد i بمقدار 0.01

السطر رقم ٧

```
sum += i ;
```

عملية الجمع وتتم بإضافة قيمة العداد i في كل مرة إلى المتغير sum

السطر رقم ٨

```
System.out.println ( " the sum is " + sum ) ;
```

يمثل جملة الطباعة التي تقوم بطباعة المتغير sum وهو عبارة عن حاصل جمع الأرقام

0.01 + 0.02 + 0.03 + ----------- + 0.1

أمثلة على استخدام جملة **for**

١ - تغير العداد من 1 إلى 100 بزيادة العداد في كل حلقة بمقدار 1
= 100 ; i + +)<for (int f = 1 ; i
٢ - تغير العداد من 7 إلى 77 بزيادة العداد في كل مرة بمقدار 7
= 77 ; i + = 7)<for (int i = 1 ; i
٣ - تغير العداد من 20 إلى 2 بإنقاص العداد في كل مرة بمقدار 2
= 2 ; i - = 2)>for (int i = 20 ; i
٤ - تغير العداد بالقيم التالية على الترتيب 2 , 5 , 8 , 11, 14 , 17 , 20
= 20 ; i + = 3) <for (int i= 2 ; i
٥ - تغير العداد بالقيم التالية على الترتيب 99 , 88 , 77 , 66 , 55 , 44 , 33 , 22 , 11
= 0 ; 0 - = 11)>for (int j = 99 ; j

خطأ شائع

وضع فاصلة فقط بدلا من الفاصلة المنقوطة التي تفصل بين أدوات التحكم في جملة for يعطي خطأ في بناء الجملة syntax error .

حلقات **for** المتداخلة

المثال التالي يستخدم الحلقات المتداخلة لطباعة جدول الضرب ، تتكون الحلقات المتداخلة من حلقة خارجية وحلقة أخرى داخلية أو أكثر ، وفي كل مرة تتكرر الحلقة الخارجية يتم تكرار الحلقات الداخلية من بداية العداد إلى نهايته.

مثال

```
1.  class TestMultable
2.  {
3.  public static void main ( string [ ]  args )
4.  {
5.  // display the title
6.  System.out.println ( "          multiplication table " ) ;
7.  System.out.println ( " --------------------------------------------- " );
8.  // display the number title
9.
10. System.out.print ( "  | " ) ;
11. for ( int  j=1 ;  j <= 9;  j ++ )
12. System.out.print ( "  " + j ) ;
13. System.out.println ( " " ) ;
14.
15. // print table body
16. for ( int  i = 1 ;  i <= 9 ;  i ++ )
17. {
18. System.out.print ( i + " | " ) ;
19. for ( int  j = 1 ;  j <= 9 ;  j ++ )
20. {
21.
22. // display the product and align properly
23.
24. if ( i * j  < 10 )
25. System.out.print ( "  " + i* j ) ;
26. else
27. System.out.print ( " " + i * j ) ;
28. }
29. System.out.println ( " " ) ;
30. }
31. }
32. }
```

```
Output Window
              multiplication table
 ---------------------------------------------
  |   1  2  3  4  5  6  7  8  9
1 |   1  2  3  4  5  6  7  8  9
2 |   2  4  6  8 10 12 14 16 18
3 |   3  6  9 12 15 18 21 24 27
4 |   4  8 12 16 20 24 28 32 36
5 |   5 10 15 20 25 30 35 40 45
6 |   6 12 18 24 30 36 42 48 54
7 |   7 14 21 28 35 42 49 56 63
8 |   8 16 24 32 40 48 56 64 72
9 |   9 18 27 36 45 54 63 72 81
Program Input:
```

شكل (3-12) مثال على الحلقات المتداخلة

شرح البرنامج

السطر رقم ٦

```
System.out.println ( "     multiplication table " ) ;
```

يقوم بطباعة العنوان multiplication table

السطر رقم ٧

```
System.out.println ( " --------------------------------------------- " );
```

يقوم بطباعة السطر الثاني وهو عبارة عن فاصل - - - - - - - - - - - - - - - - - -

السطور ١١ إلى ١٢

```
for ( int   j=1 ;  j <= 9;  j ++ )
System.out.print ( "  " + j ) ;
```

يقوم بطباعة الأرقام من 1 إلى 9 وذلك في السطر الثالث

السطور ١٦ إلى ٣٠

عبارة على حلقة for متداخلة ، الحلقة الخارجية لها عداد i والحلقة الداخلية لها عداد j .

لاحظ أنه لكل i جديدة يعرض حاصل ضرب j*i ويتم ذلك في الحلقة الداخلية حيث قيم j تتراوح من 1 إلى 9

السطر رقم ٢٤

```
if ( i*j   <  10 )
```

عبارة عن جملة if الشرطية وهى تبحث ما إذا كان حاصل ضرب i وj أقل من 10 أم لا وهذا الشرط يفيد في عمل محاذاة للأرقام أثناء الطباعة ، فإذا تحقق الشرط نفذ البرنامج للسطر التالي مباشرة وهو

```
System.out.print ( "  " + i*j ) ;
```

وهذا السطر يقوم بطباعة مسافتين قبل طباعة حاصل ضرب أل i*j .

أما إذا لم يتحقق الشرط وكان حاصل الضرب i*j أكبر من أو يساوي 10

معنى ذلك أن العدد الناتج يأخذ خانتين في الطباعة لذلك تنفذ جملة else

```
System.out.print ( " " + i*j ) ;
```

لاحظ طباعة مسافة واحدة قبل طباعة حاصل الضرب

السطر رقم ٢٩

```
System.out.println ( " " ) ;
```

لطباعة سطر خالٍ في نهاية كل سطر

جمل continue و break

تستعمل هذه الجمل عندما يراد تغيير المسار الطبيعي للبرنامج فمثلا عندما تستخدم جملة break داخل بناء جملة switch , do/while , for , while تسبب الخروج منها فورا ويستمر تنفيذ باقي جمل البرنامج التي تلي بناء الجملة والاستخدام الشائع لجملة break هو للهروب مبكرا من تنفيذ حلقة أو لإهمال تنفيذ باقي جملة switch .

وفي المثال التالي سوف نقوم بتوضيح عمل جملة break

مثال

```
1.    // Fig. 3.13: BreakTest.java
2.    // Using the break statement in a for structure
3.
4.    // Java extension packages
5.    import javax.swing.JOptionPane;
6.
7.    public class BreakTest {
8.  // main method begins execution of Java application
9.  public static void main( String args[] )
10. {
11. String output = "";
12. int count;
13.
14. // loop 10 times
15. for ( count = 1; count <= 10; count++ ) {
16.
17. // if count is 5, terminate loop
18. if ( count == 5 )
19. break;  // break loop only if count == 5
20.
21. output += count + " ";
22.
23. }  // end for structure
24.
25. output += "\nBroke out of loop at count = " + count;
26. JOptionPane.showMessageDialog( null, output );
27.
28. System.exit( 0 );  // terminate application
29.
30. }  // end method main
31.
32. }  // end class BreakTest
```


شكل (3-13) مثال على استخدام جملة break

شرح البرنامج :

في البرنامج السابق لاحظ وجود جملة if في السطر رقم 18 وهى موجودة داخل بناء جملة for وشرط جملة if أن يكون المتغير count يساوي 5 فإذا تحقق هذا الشرط نفذت الجملة التالية رقم 19 وفيها نرى جملة ; break ، هذه الجملة تتسبب في إنهاء الحلقة التكرارية والخروج منها لينفذ البرنامج بعدها مباشرة أول جملة بعد الحلقة وهى في السطر رقم 25

```
output + =  " in broke out of  loop at count = " + count ;
```

لاحظ إضافة النص السابق إلى المتغير output بالإضافة إلى قيمة المتغير count وذلك قبل طباعته باستخدام صندوق رسالة في السطر رقم 26

```
JOptionPane.showMessageDialog ( null , output ) ;
```

وفي المثال التالي سوف نقوم بتوضيح عمل جملة continue ولنرى تأثير جملة continue والفرق بينها وبين جملة break نقوم بكتابة نفس البرنامج السابق ولكن نستبدل جملة break بجملة continue نلاحظ أن جملة continue تتعدى الجملة الباقية في الحلقة لتبدأ تنفيذ الحلقة من البداية بالقيمة التالية للعداد .

ففي المثال إذا تحقق الشرط وكانت قيمة count تساوي 5 قام البرنامج بتنفيذ جملة continue والتي تجعل البرنامج يهمل باقي الجمل داخل الحلقة وهى في المثال السطر رقم 29

```
Output + = count + " " ;
```

ثم يعود لتنفيذ الحلقة من البداية بالقيمة التالية للعداد count وهى 6

ويكون شكل تنفيذ البرنامج باستخدام جملة continue بدلا من جملة break على الشكل التالي

مثال

```
1.  // Fig. 3.14: ContinueTest.java
2.  // Using the continue statement in a for structure
3.  // Java extension packages
4.  import javax.swing.JOptionPane;
5.
6.  public class ContinueTest {
7.
8.  // main method begins execution of Java application
9.  public static void main( String args[] )
10. {
11. String output = "";
12.
13. // loop 10 times
14. for ( int count = 1; count <= 10; count++ ) {
15. // if count is 5, continue with next iteration of loop
16. if ( count == 5 )
17. continue;  // skip remaining code in loop
18. // only if count == 5
19. output += count + " ";
20. }  // end for structure
21. output += "\nUsed continue to skip printing 5";
22. JOptionPane.showMessageDialog( null, output );

23. System.exit( 0 );  // terminate application

24. } // end method main

25. }  // end class ContinueTest
```

Message

1 2 3 4 6 7 8 9 10
Used continue to skip printing 5

OK

شكل (3-14) مثال على استخدام جملة continue

جمل **break** و **continue** المعنونة

تعرضنا سابقا لجملة break وقلنا أنها تتسبب في الهروب من بناء الجملة التي تشتمل عليها فقط سواء كان هذا البناء هو جملة while أو for أو do/while أو switch

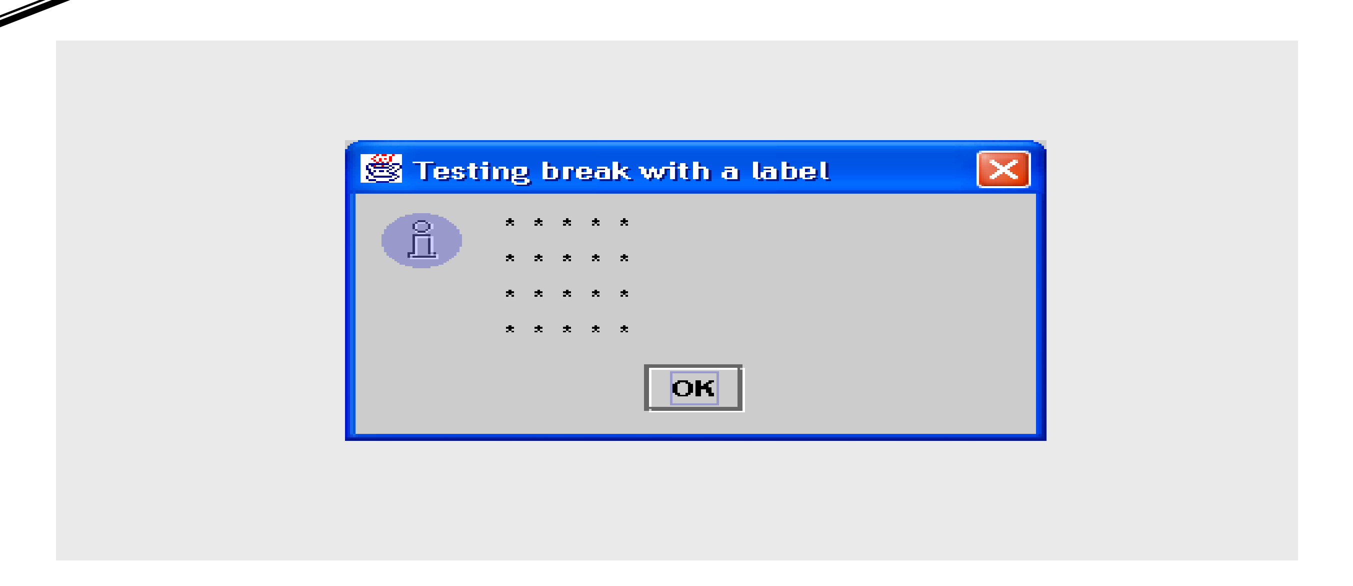
ولكي نستطيع عمل هروب من مجموعة من بناءات الجمل معا لابد أن نستخدم جملة الهروب المعنونة labeled break ، عندما تستخدم هذه الجملة وتنفذ داخل بناء while أو for أو do/while أو switch تتسبب في الهروب فورا من هذا البناء وأي عدد آخر من البناءات التي تشتمل عليها ، ثم بعد ذلك يستأنف تنفيذ البرنامج بعد القالب المعنون (القالب المعنون هو عبارة عن مجموعة من الجمل داخل البرنامج والتي تكون محصورة بين قوسين وفي بدايتها عنوان)

```
Stop : {
            // جمل القالب
       }
```

تستخدم عادة جملة الهروب المعنونة labeled break للخروج من الحلقات المتداخلة والتي يمكن أن تكون بواسطة while أو for أو do/while أو switch

مثال

```
1.  // Fig. 3.15: BreakLabelTest.java
2.  // Using the break statement with a label
3.
4.  // Java extension packages
5.  import javax.swing.JOptionPane;
6.
7.  public class BreakLabelTest {
8.
9.  // main method begins execution of Java application
10. public static void main( String args[] )
11. {
12. String output = "";
13.
14. stop: {   // labeled block
15.
16. // count 10 rows
17. for ( int row = 1; row <= 10; row++ ) {
18. // count 5 columns
19. for ( int column = 1; column <= 5 ; column++ ) {
20.
21. // if row is 5, jump to end of "stop" block
22. if ( row == 5 )
23. break stop; // jump to end of stop block
24.
25. output += "*  ";
26.
27. }  // end inner for structure
28.
29. output += "\n";
30.
31. }  // end outer for structure
32.
33. // the following line is skipped
34. output += "\nLoops terminated normally";
35.
36. }  // end labeled block
37.
38. JOptionPane.showMessageDialog(
39. null, output,"Testing break with a label",
40. JOptionPane.INFORMATION_MESSAGE );
41.
42. System.exit( 0 );  // terminate application
43.
44. }  // end method main
45.
46. }  // end class BreakLabelTest
```



شكل (3-15) مثال على استخدام جملة break المعنونة

شرح البرنامج:

هذا المثال يوضح استخدام جملة الهروب المعنونة مع الحلقات المتداخلة نلاحظ من البرنامج أن القالب في السطور من 14 إلى 36 كما أنه يبدأ بعنوان (دائما ما يكون العنوان عبارة عن معرف متبوع بـ :)

سطر 14

```
stop: {   // labeled block
```

بداية القالب واسم العنوان

السطور ١٧ - ٣١

عبارة عن حلقات for المتداخلة

السطر ٣٤

```
output += "\nLoops terminated normally";
```

إضافة النص Loop terminated normally إلى المتغير output.

نلاحظ أنه عندما تكون قيمة المتغير row تساوي 5 أي يتحقق الشرط الموجود في السطر رقم ٢٢ يتم

تنفيذ جملة الهروب المعنونة الموجودة في السطر رقم ٢٣

```
if ( row == 5 )
break stop;  // jump to end of stop block
```

هذه الجملة تنهي عمل كل من بناء for الموجودة في السطر رقم ١٩ وبناء جملة for الخارجية والموجودة في السطر رقم ١٧ ويستأنف تنفيذ بقية البرنامج بداية من السطر رقم ٣٨ أي أول سطر بعد القالب المعنون.

ملحوظة: بناء جملة for الخارجية يتم تنفيذ ما بها من جمل 4 مرات فقط (حتى تصل قيمة المتغير row إلى 5) لذلك السطر رقم ٣٤ لا ينفذ أبدا وذلك لأنه داخل القالب المعنون وجملة for الخارجية لا تكتمل أبدا .

كما قلنا سابقا فإن جملة continue المعنونة تهمل تنفيذ باقي الجمل في الحلقة لتبدأ تنفيذ الحلقة من البداية بالقيمة التالية للعداد أما جملة continue المعنونة labeled continue فهي تتسبب في إهمال باقي الجمل في الحلقة وأي حلقات أخرى تشتمل عليها ، ثم تبدأ بتنفيذ بناء التكرار المعنون structure labeled repetition الذي يشملها وذلك بقيمة جديدة للعداد في كل الحلقات.

بناء التكرار المعنون هو حلقة تكرارية تبدأ بعنوان.

مثال

```
1.  // Fig. 3.16: ContinueLabelTest.java
2.  // Using the continue statement with a label
3.
4.  // Java extension packages
5.  import javax.swing.JOptionPane;
6.
7.  public class ContinueLabelTest {
8.
9.  // main method begins execution of Java application
10. public static void main( String args[] )
11. {
12. String output = "";
13.
14. nextRow:   // target label of continue statement
15.
16. // count 5 rows
17. for ( int row = 1; row <= 5; row++ ) {
18. output += "\n";
19.
20. // count 10 columns per row
21. for ( int column = 1; column <= 10; column++ ) {
22.
23. // if column greater than row, start next row
24. if ( column > row )
```

```
25. continue nextRow; // next iteration of
26.
27. // labeled loop
28.
29. output += "*  ";
30.
31. }  // end inner for structure
32.
33. }  // end outer for structure
34.
35. JOptionPane.showMessageDialog(
36. null, output,"Testing continue with a label",
37. JOptionPane.INFORMATION_MESSAGE );
38.
39. System.exit( 0 );  // terminate application
40.
41. }  // end method main
42.
43. }  // end class ContinueLabelTest
```

```
Testing continue with a label
*
* *
* * *
* * * *
* * * * *
OK
```

شكل (3-16) مثال على استخدام جملة continueالمعنونة

أسئلة وتمارين على التحكم البنائي

١ ضع علامة صح أمام العبارة الصحيحة وعلامة خطأ أمام العبارة الخاطئة لكل من الجمل التالية

أ - لابد من وجود الحالة الافتراضية default داخل بناء switch

ب – لابد من وجود جملة break بعد الحالة الافتراضية default داخل بناء switch

ج _ التعبير (x > y && a < b) صحيحاً إذا كان x > y صحيحاً أو a < b صحيح

د _ يقال التعبير يحتوي على العامل ׀׀ انه صحيح إذا كان أحد المعاملات صحيحاً أو كلاهما معا .

٢ اكتب جملة أو عدة جمل بلغة الجافا لكي تقوم بعمل المهام التالية

أ _ جمع الأعداد الفردية من 1 إلى 99 باستخدام حلقة for ، افرض أن المتغيرات الصحيحة sum Count قد تم تعريفها .

ب _ طباعة الأعداد الصحيحة من 1 إلى 20 باستخدام حلقة while
افرض أن متغير العدد هو x قد تم تعريفه ولكن لم يعطَ له القيمة الابتدائية
اطبع 5 أعداد فقط في كل سطر
ملحوظة : استخدم x % 5 إذا كان ناتج التعبير السابق يساوي ٥
اطبع سطراً جديداً وإذا لم يكن يساوي صفراً يتم طباعة مسافة فقط

ج _ كرر السؤال السابق ولكن باستخدام حلقة for

٣ - وضح ناتج تنفيذ هذا البرنامج ؟

```
1  public class printing  {
2
3           public static void main ( String args [ ] )
4            {
5               for ( int  i = 1; i  <= 10;  i++ ) {
6
7                   for ( int  j = 1; j  <= 5;  j++ )
8                       System.out.print ('@' );
9
10                  System.out.println ( );
11
12             }
13
14         }
15
16    }
```

4 - ما العمل الذي يقوم به هذا الجزء من البرنامج ؟

```
        for  (  i = 1;  i  <= 5;  i++ ) {

            for  (  j = 1;  j  <= 3;  j++ ) {

                for  (  k = 1;  k  <= 4;  k++ ) {
                    System.out.print ( ' * ' );

                System.out.println ( );
        }

        System.out.println ( );
    }
```

٥ - اكتب برنامجاً يوجد الرقم الأصغر لمجموعة من الأرقام الصحيحة المُدخلة بواسطة المُستخدم، افرض أن الرقم الأول يمثل عدد الأرقام.

٦ - اكتب برنامج يقوم بحساب حاصل ضرب الأعداد الفردية من 1 إلى 15، ثم يقوم بعرض الناتج في صندوق رسالة.

٧ - يستخدم المضروب في كثير من المسائل الرياضية، ومضروب العدد 8 (يكتب بالشكل !8 ويُقال له مضروب 8) والمضروب هو عبارة عن حاصل ضرب الأعداد الصحيحة الموجبة من 1إلى 8.
اكتب برنامجاً يقوم بحساب مضروب الأعداد الصحيحة من 1 إلى 5، واعرض الناتج داخل صندوق رسالة.

٨ - باستخدام الحلقات المُتداخلة، اكتب برنامجاً يقوم بعرض كلٍ من الأشكال التالية:

- ملحوظة: برنامج لكل شكل

(a)
```
         *
        **
       ***
      ****
     *****
    ******
   *******
  ********
 *********
**********
```

(b)
```
**********
 *********
  ********
   *******
    ******
     *****
      ****
       ***
        **
         *
```

(c)
```
**********
*********
********
*******
******
*****
****
***
**
*
```

(d)
```
*
**
***
****
*****
******
*******
********
*********
**********
```

٩ - اكتب برنامج يقوم بحساب مجموع المتوالية غير المنتهية

$$\pi = 4 - \frac{4}{7} - \frac{4}{5} + \frac{4}{3} + \frac{4}{9} - \frac{4}{11} + ...$$

اطبع جدولاً به قيم π مُقربة باستخدام حد واحد من المتوالية السابقة، ثم باستخدام حدين من المتوالية، ثم باستخدام ثلاثة حدود، ثم أوجد عدد الحدود المستخدمة لتكون π تساوي 3.14159

١٠- وضح ناتج تنفيذ هذا البرنامج ؟

```
1  public class Mystery2  {
2
3          public static void main ( string args [ ] )
4           {
5              int count  = 1;
6
7             while  (  count <= 10  )  {
8                     System.out.println (
9                            count %  2 = = 1 ? "*****"   :   "+++++++++" );
10                      ++count;
11                }
12          }
13     }
```

١١- وضح ناتج تنفيذ هذا البرنامج ؟

```
1  public class Mystery3  {
2
3          public static void main ( String args [ ] )
4           {
5              int row  = 10,  column;
6
7             while  (  row >= 1  )  {
8                 column = 1;
9
10                while  (  column <= 10  )  {
11                    System.out.print (row %  2 = = 1 ? "<"  :  ">" );
12                    ++column;
13                 }
14
15               --row;
16               System.out.println( );
17           }
18       }
19     }
```

ملحـــق أ

لكي نقوم بكتابة برنامج بلغة الجافا ثم تنفيذه لابد لنا من وجود :

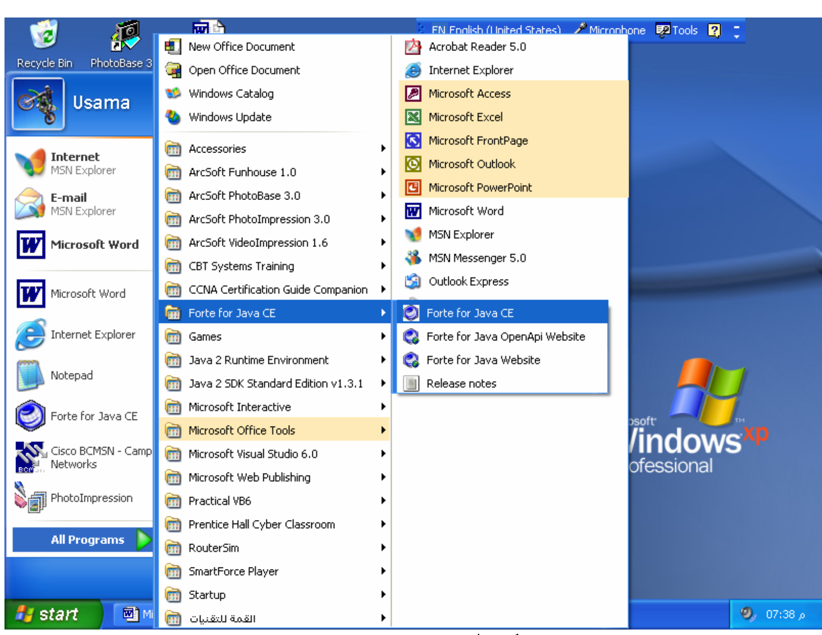
أولا : Java 2 Software Development Kit والمعروفة اختصارا بـ JDK وموجود منها الآن الإصدار رقم 1.4 ، وهي عبارة عن تعليمات اللغة نفسها.

ثانيا : Integrated Development Environment والمعروفة اختصارا بـ IDE وهي عبارة عن البيئة التي نكتب فيه البرنامج أو المحرر .

برنامج الـ Forte هو أحد البرامج التي أنتجتها وطورتها شركة صن مايكروسيستمز Sun Microsystems لكي يستخدمه مبرمجو لغة الجافا في تطوير البرامج (تصميم وكتابة وترجمة ثم تنفيذ) أي هو عبارة عن IDE ، لذلك لابد قبل تحميل هذا البرنامج أن نحمل الـ JDK ثم بعد ذلك نقوم بتحميل برنامج الـ Forte ، وأثناء عملية التحميل يطلب منا أن نحدد مسار الـ JDK .

وسوف نتعرض الآن لكيفية كتابة برنامج بسيط بلغة الجافا بواسطة برنامج Forte ومن ثم عمل ترجمة له ثم تنفيذه

١ - تشغيل برنامج الــForte



شكل (A-1) طريقة تشغيل البرنامج

٢ - اختيار New file

عند بدء التشغيل تظهر الشاشة المقابلة شكل (A-2) فنختار منها New كما في الشكل وذلك لإنشاء ملف جديد.



شكل (A-2)

٢ - اختيار النموذج Template

تظهر بعد ذلك شكل(A-3) المقابل لنختار منها نوع البرنامج المطلوب عمله، وتفيد هذه الطريقة في أن البرنامج المختار يتم فتحه وكتابة الأشياء الأساسية

به مثل تعريف الفصل ، بداية ونهاية البرنامج ، ... الخ

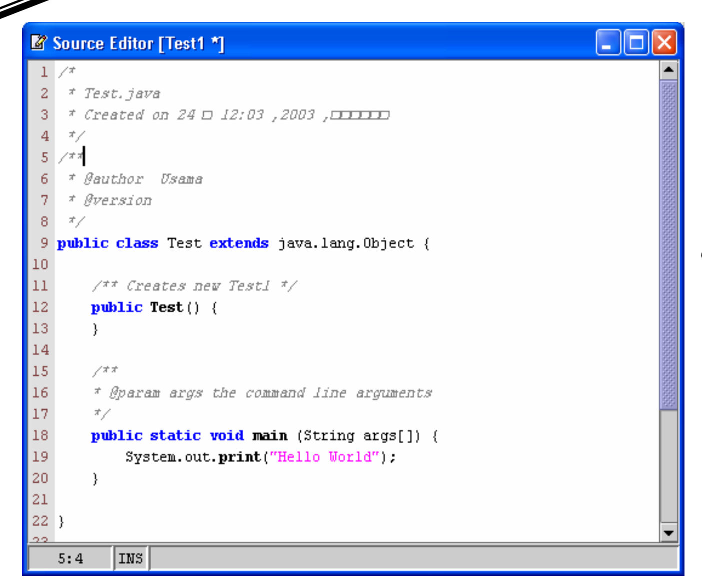
على سبيل المثال نختار Classes ثم من القائمة المسدلة نختار Main إذا كنا نريد عمل برنامج تطبيق .

في خانة Name نكتب اسم الفصل

شكل (A-3)

نلاحظ وجود بعض الاختيارات الأخرى إذا ضغطنا على زر Next لن نتعرض اليها الآن ولكننا سنضغط زر Finish مباشرة وتظهر الشاشة التالية شكل (A-4) التي تسأل المستخدم إذا كان يريد إضافة هذا الفصل إلى المشروع لكي يظهر الفصل في مستكشف المشروع وتكون إجابتنا بنعم Yes

شكل (A-4)



شكل (A-5)

٣ - كتابة البرنامج

تظهر لنا الشاشة المقابلة شكل (A-5) لنقوم بكتابة البرنامج ونلاحظ وجود تعريف الفصل بالاسم الذي اخترناه من قبل بالإضافة لوجود الطريقة Main التي لابد من كتابتها إذا كنا نقوم بعمل تطبيق.

نقوم بكتابة البرنامج داخل الطريقةMain

٤ - عمل ترجمة Compilation

بعد كتابة البرنامج يتم عمل ترجمة له كما بالشكل(A-6) من قائمة Build نختار Compile أو F9 أو من شريط الأدوات كما بالشكل يتم عمل ترجمة للبرنامج فإذا كانت هناك أخطاء نقرأ الجملة التالية بجانب شريط الأدوات

شكل (A-6)

Error Compiling Test حيث Test هو اسم الفصل وتظهر لنا شاشة الخرج وبها تعريف الخطأ ورقم السطر الموجود به بالإضافة إلى تحديد السطر داخل البرنامج بلون مختلف. شكل (A-6)

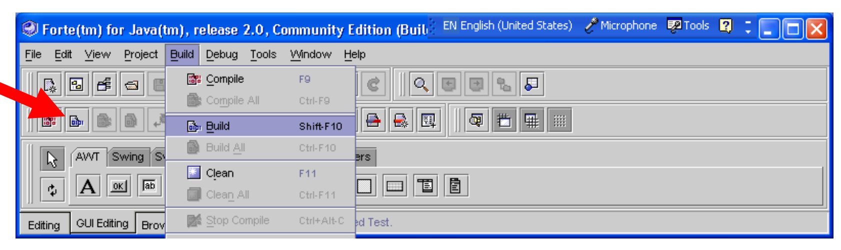
لاحظ أن الخطأ في المثال هو نسيان الفاصلة المنقوطة وحدد الخطأ في السطر رقم ٢٤ وأن السطر ٢٤ هو السطر التالي للسطر الذي وجد به الخطأ لاحظ أيضا تحديد السطر باللون الأخضر في شاشة البرنامج المصدر.

نقوم بعد ذلك بتصحيح الخطأ ثم نعمل ترجمة مرة أخرى هذه المر تظهر الجملة التالية بجوار

شكل (A-6)

شريط الأدوات Finished Test وهي تعني أن البرنامج تم عمل له ترجمة وتم إنشاء ملف بنفس الاسم ولكن بامتداد class.

بعد ذلك نقوم بعمل Build للبرنامج من قائمة Build نختار Build أو نضغط Shift+f10 أو من شريط الأدوات كما بالشكل(A-7)



٥ - التنفيذ

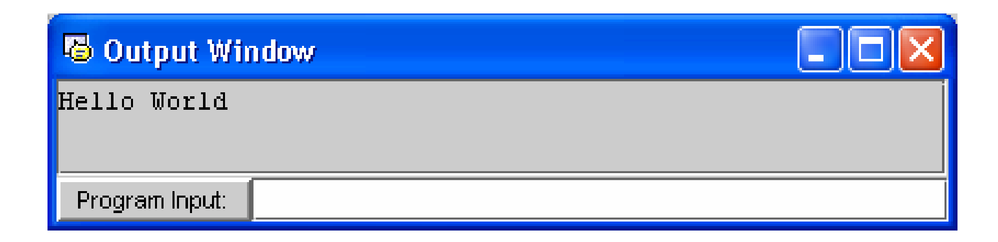
شكل (A-7)

يتم عمل تنفيذ للبرنامج من خلال اختيار Execute من قائمة Build أو F6 أو من خلال شريط الأدوات



شكل(A-8)

يظهر ناتج تنفيذ البرنامج في شاشة الخرج Output Window كما بالشكل (A-9)

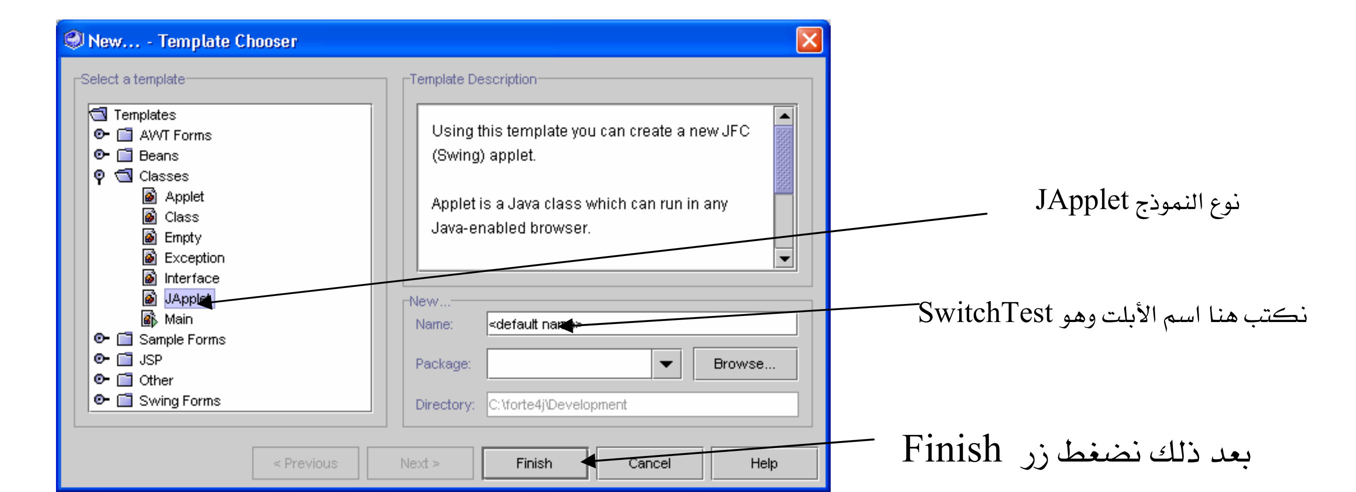


شكل(A-9)

مثال

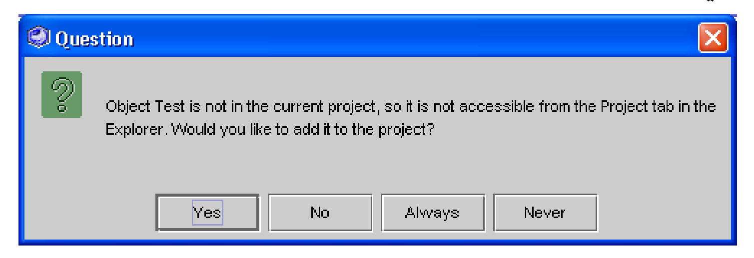
في هذا المثال نقوم باستخدام نموذج آخر لكتابة البرنامج وهو النموذج JApplet الموجود تحت القائمة Classes

اختيار النموذج انظر الشكل(A-10)



شكل (A-10)

وتظهر الشاشة التالية شكل(A-11) التي تسأل المستخدم اذا كان يريد إضافة هذا الفصل إلى المشروع لكي يظهر الفصل في مستكشف المشروع وتكون إجابتنا بنعم Yes



شكل (A-11)

نقوم بكتابة البرنامج في نافذة المحرر كما بشكل (A-12)

Source Editor [SwitchTest *]

```
1  /*
2   * SwitchTest.java
3  */
4  public class SwitchTest extends javax.swing.JApplet {
5
6      /** Creates new SwitchTest */
7      public SwitchTest() {
8      }
9      int choice;    // user's choice of which shape to draw
10
11    // initialize applet by obtaining user's choice
12    public void init()
13    {
14       String input;   // user's input
15
16       // obtain user's choice
17       input = JOptionPane.showInputDialog(
18                 "Enter 1 to draw lines\n" +
19                 "Enter 2 to draw rectangles\n" +
20                 "Enter 3 to draw ovals\n" );
21
```

3:2　INS

Test1 *　SwitchTest *

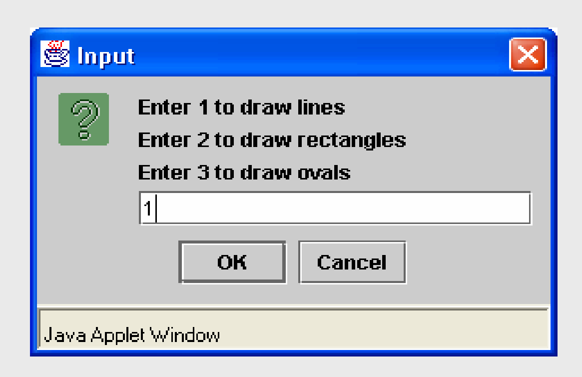
شكل(A-12)

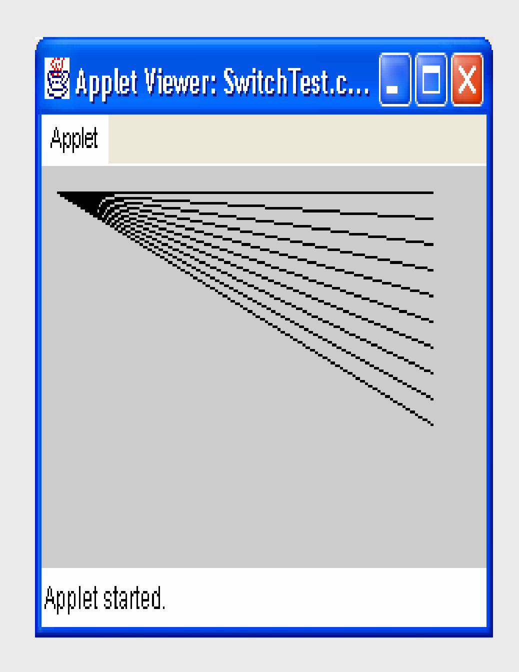
بعد كتابة البرنامج نقوم بعمل ترجمة له compile ثم بعد ذلك Build ثم بعد ذلك نقوم بالتنفيذ ونلاحظ أنه عند تنفيذ هذا البرنامج وهو أبلت يقوم برنامج الـ Forte بإنشاء ملف HTML ثم يقوم بتحميل الأبلت عليه ومن ثم تنفيذه من خلال عارض الأبلت appletviewer

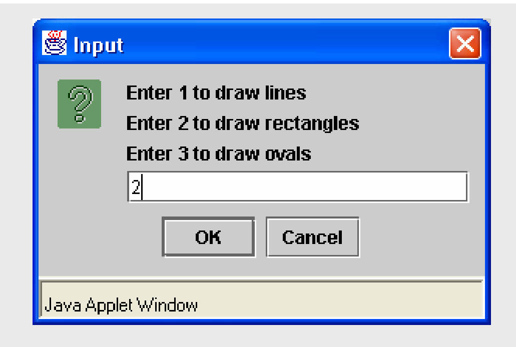
الشكل (A-13) يوضح البرنامج كاملا بالإضافة الى شكل التنفيذ:

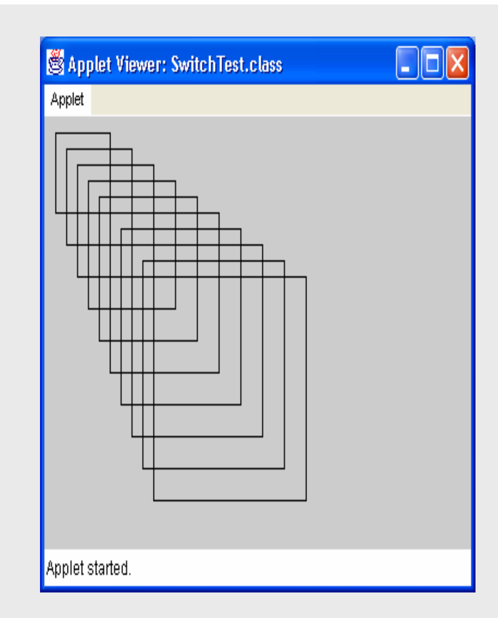
```
1.  // Fig. A.13: SwitchTest.java
2.  // Drawing lines, rectangles or ovals based on user input.
3.
4.  // Java core packages
5.  import java.awt.Graphics;
6.
7.  // Java extension packages
8.  import javax.swing.*;
9.
10. public class SwitchTest extends JApplet {
11. int choice;   // user's choice of which shape to draw
12.
13. // initialize applet by obtaining user's choice
14. public void init()
15. {
16. String input;  // user's input
17.
18. // obtain user's choice
19. input = JOptionPane.showInputDialog(
20. "Enter 1 to draw lines\n" +
21. "Enter 2 to draw rectangles\n" +
22. "Enter 3 to draw ovals\n" );
23.
24. // convert user's input to an int
25. choice = Integer.parseInt( input );
26. }
27.
28. // draw shapes on applet's background
29. public void paint( Graphics g )
30. {
31. // call inherited version of method paint
32. super.paint( g );
33.
34. // loop 10 times, counting from 0 through 9
35. for ( int i = 0; i < 10; i++ ) {
36.
37. // determine shape to draw based on user's choice
38. switch ( choice ) {
39.
40. case 1:
```

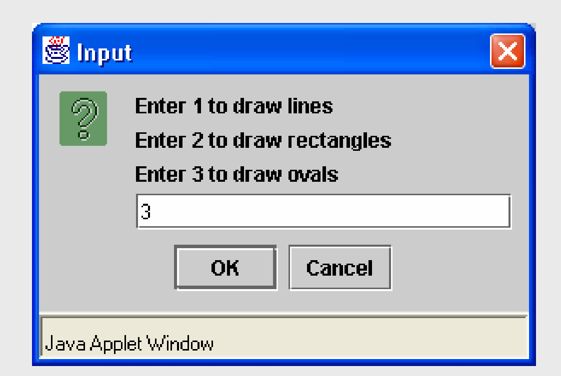
```
41. g.drawLine( 10, 10, 250, 10 + i * 10 );
42. break;  // done processing case
43.
44. case 2:
45. g.drawRect( 10 + i * 10, 10 + i * 10,
46. 50 + i * 10, 50 + i * 10 );
47. break;  // done processing case
48. case 3
49. g.drawOval( 10 + i * 10, 10 + i * 10,
50. 50 + i * 10, 50 + i * 10 );
51. break;  // done processing case
52. default:
53. g.drawString( "Invalid value entered",
54. 10, 20 + i * 15 );
55.
56. }  // end switch structure
57.
58. }  // end for structure
59.
60. }  // end paint method
61.
62. }  // end class SwitchTest
```

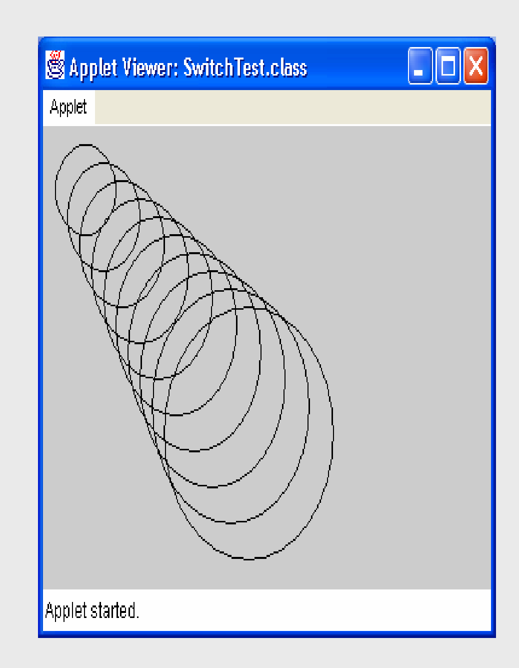












شكل (A-13) برنامج أبلت باستخدام برنامج الفورتي

ملحق ب

برامج الأبلت Applets

تعرضنا من خلال الوحدات السابقة إلى نوع من أنواع البرامج في لغة الجافا وهو التطبيقApplication وقلنا إنه يوجد نوع آخر من البرامج ألا وهو الأبلت Applet وتتميز هذه البرامج بإمكانية إدماجها داخل صفحات الويب، فمثلا عندما يتم تحميل صفحة ويب تحتوي على أبلت من خلال المتصفح فيقوم هذا المتصفح بتحميل الأبلت ويبدأ بتنفيذه.

متصفح الويب الذي يقوم بتنفيذ الأبلت يسمى حاوي الأبلت Applet container، تحتوي حزمة تطوير البرامج بالجافا Java 2 Software Development Kit على حاوي أبلت يسمى عارض الأبلت Applet Viewer وهو يستخدم لعمل اختبار للأبلت قبل دمجها مع صفحة الويب.

يوجد العديد من المتصفحات لا تدعم الجافا مباشرة مثل متصفح مايكروسوفت، يعتبر متصفح Netscape 6 أحد المتصفحات التي تدعم الجافا.

ملحوظة: لتنفيذ الأبلت على أحد المتصفحات التي لا تدعم الجافا نستخدم **Java Plug-in (Converter)** وسنتعرض له لاحقا.

مثال

```
1.  // Fig. B.1: WelcomeApplet.java
2.  // A first applet in Java.
3.
4.  // Java core packages
5.  import java.awt.Graphics;    // import class Graphics
6.
7.  // Java extension packages
8.  import javax.swing.JApplet;  // import class JApplet
9.
10. public class WelcomeApplet extends JApplet {
11.
12. // draw text on applet's background
13. public void paint( Graphics g )
14. {
15. // call inherited version of method paint
16. super.paint( g );
17.
18. // draw a String at x-coordinate 25 and y-coordinate 25
19. g.drawString( "Welcome to Java Programming!", 25, 25 );
20.
21. }  // end method paint
22.
23. }  // end class WelcomeApplet
```



شكل (B.1) برنامج أبلت وشكل التنفيذ

يوضح هذا البرنامج العديد من الخصائص الهامة للجافا ، لاحظ أن السطر رقم 19 هو الذي يقوم بالعمل الفعلي للبرنامج وهو رسم النص التالي على الشاشة

Welcome to Java Programming!

شرح البرنامج

السطور 1 – 2

```
// Fig. B.1: WelcomeApplet.java
// A first applet in Java.
```

كما قلنا سابقا أي سطر يبدأ بـ // يعتبر ملاحظة أي لا يدخل ضمن البرنامج ولكن يستخدم للتوضيح للمبرمج وهنا السطر الأول يوضح اسم البرنامج ورقم الشكل كما أن السطر الثاني يوضح الهدف من البرنامج.

السطر رقم 5

```
import java.awt.Graphics;    // import class Graphics
```

قلنا سابقا إن لغة الجافا تحتوي على مكونات معرفة سابقا تسمى فصول classes وهذه الفصول مجمعة داخل حزم packages . والسطر رقم 5 هو عبارة عن جملة import التي تقول للمترجم أن يحمل الفصل Graphics من الحزمة java.awt . الفصل Graphics يسمح للأبلت أن تقوم برسم أشكال مثل خط ، مستطيل ، شكل دائري ، سلسلة من الحروف ، ...الخ.

السطر رقم 8

```
Import javax.swing.JApplet; // import class JApplet
```

هو أيضا عبارة عن جملة import والتي تخبر المترجم أن يقوم بتحميل الفصل JApplet من الحزمة javax.swing . نقوم بدمج هذا الفصل عادة عندما تقوم بانشاء أبلت.
ملحوظة: يوجد إصدار قديم من هذا الفصل يسمى Appletوموجود في الحزمة java.applet

كما هو الحال في برامج التطبيقات فإن كل أبلت تحتوي على الأقل على تعريف لفصل واحد وهذا الفصل لابد وأن يكون امتداد لفصل آخر موجود من قبل بمعنى أن الفصل لا ينشأ من الصفر ولكن ينشأ كتكملة وامتداد لفصل آخر وذلك نراه في تعريف الفصل في الكلمة extends ثم يتبعها اسم الفصل الأساسي ولكن إذا لم نكتب هذه الكلمة ثم اسم الفصل الأساسي وذلك في برامج التطبيقات اعتبر المترجم ضمنيا أن الفصل امتداد للفصل Object أما في الأبلت فيجب كتابة هذه الكلمة ويتبعها اسم الفصل JavaApplet كما في السطر رقم 10

```
Public class WelcomeApplet extends JApplet {
```

وهو تعريف الفصل WelcomeApplet . في نهاية هذا السطر يوجد القوس الأيسر } والقوس الأيمن له موجود في السطر رقم 23 وبينهما توجد تعليمات الفصل
في هذه العلاقة الوراثية يسمى الفصل JApplet بالفصل السوبر أو الأساسيSuperclass كما يسمى الفصل WelcomeApplet بالفصل الفرعي subclass ، يرث الفصل الفرعي كل خصائص الفصل السوبر كما يرث أيضا جميع الطرق التي به بالإضافة الخصائص والطرق الخاصة بالفصل الفرعي نفسه وهي على سبيل المثال قدرة الفصل WelcomApplet على رسم النص Welcome To Java Programming ! على الأبلت.
سؤال : لماذا دائما الفصل أبلت يكون امتداد لفصل آخر وهو JApplet ؟
الإجابة: لكي يقوم عارض الأبلت أو المتصفح بعرض الأبلت يحتاج على الأقل إلى 200 طريقة ونلاحظ في المثال السابق أنه يحتوي على طريقة واحدة فقط وذلك لأنه ورث الطرق الأخرى من الفصل السوبر، فإذا كنا في كل مرة نكتب أبلت نحتاج لعمل 200 طريقة فإننا لن نعمل أبلت أبدا.

السطر رقم 13

```
Public void paint( Graphics g )
```

هـذا السـطر يبـدأ بتعريـف الطريقـة paint وهـي واحـدة مـن ثـلاث طـرق أخـرى يقـوم حـاوي الأبلـت باسـتدعائهم عنـد تنفيـذ الأبلـت وهـم (init , start , paint) وهـذه الطـرق الـثلاث تـورث مـن قبـل الفصل السوبر الى الفصـل الفرعـي ، إذا لم تقـم بتعريـف أحـد هـذه الطـرق مـرة أخـرى في الأبلـت يقـوم حاوي الأبلت باستدعاء النسخة الموروثة .

ملحوظة: النسخة الموروثة من الطريقة init والطريقة start لا تحتوي على تعليمات لذلك فهـي لا تقـوم بأي مهمة كما أن النسخة الموروثة من الطريقة paint لا تقوم بعرض أي رسوم على الأبلت .

لكي نجعل الأبلت تقوم بعرض رسوم فإننا نقوم بإعادة تعريف الطريقة paint وإضافة لها جملة الرسم

السطور 13 - 14

تحتوي علـى تعريـف الطريقـة paint وتعليماتهـا ، وكمـا هـو الحـال عنـد عـرض صـندوق رسـالة فإننـا كنا نقوم باستدعاء الطريقة showMessageDialog والموجودة في الفصل JoptionPane فإننـا هنا لكي نقوم بعرض رسوم على الأبلت نسـتدعي الطريقـة paint ولكـن المبرمج لا يقـوم باسـتدعائها صراحة ولكن حاوي الأبلت هو الذي يقوم باستدعائها لكـي تجعـل الأبلـت يعـرض رسـوماً ويقـوم حـاوي الأبلـت أيضـا بتمريـر المعلومـات الـتي تحتاجهـا في الرسـم وهـي الهـدف Graphics ويسـمى بــ g ، تستخدم الطريقة paint الهدف Graphics لكي تقوم برسم الأشكال والرسوم على الأبلت .

لاحظ أن الطريقة معرفة على أنها public لكـي يسـتطيع حـاوي البلـت اسـتدعاء الطريقـة paint لذلك يجب أن تكون كل الطرق public

السطر رقم 16

```
super.paint( g );
```

هذا السطر يقوم باستدعاء النسخة الأصلية الموجودة في الفصل السوبر JApplet

السطر رقم 19

```
g.drawString( "Welcome to Java Programming!", 25, 25 );
```

كما قلنا سابقا إن هذا السطر هو الذي يقوم فعليا برسم النص على الأبلت فهو يستدعي الطريقة drawstring والموجودة داخل الهدف Graphics المسمى بـ g لذلك فإننا نستدعيها بأن يكتب اسم الهدف يتبعها اسم الطريقة وتفصلهما نقطة.

أول عنصر داخل الطريقةdrawString هو النص نفسه وهو
Welcome To Java Progamming !
ثاني عنصر هو إحداثي المحور السيني و إحداثي المحور الصادي الذي سوف نبدأ منهما الرسم على الأبلت وهما في المثال 25, 25 مع ملاحظة أن الاحداثي 0 , 0 يبدأ عند الركن العلوي في اليسار.

بعد عملية الترجمة compilation وقبل أن نستطيع تنفيذ الأبلت لابد أولا من إنشاء ملف HTML لكي يقوم بتحميل الأبلت إلى حاوي الأبلت وهو إما أن يكون المتصفح أو عارض الأبلت appletviewer وملف الـ HTML يكون له امتداد html. أو htm. ولكي نقوم بالتنفيذ لابد أن يشير ملف الـ HTML الى اسم الأبلت . والمثال (B-2) يوضح كيفية تعريف اسم الأبلت داخل ملف الـ HTML .

```
1.      <html>
2.      <applet code = "WelcomeApplet.class" width = "300" height = "45">
3.      </applet>
4.      </html>
```

شكل (B-2) ملفHTML وبه اسم الأبلت

لاحظ أن السطر رقم 2 معرف فيه اسم الفصل الأبلت وهو WelcomeApplet.class كما هو محدد في عرض وارتفاع الأبلت التي ستظهر في المتصفح (حاوي الأبلت)

ملحوظة: معظم برامج التحرير للجافا تقوم هي بإنشاء ملف الـ HTML نيابة عن المبرمج وذلك عند التنفيذ.

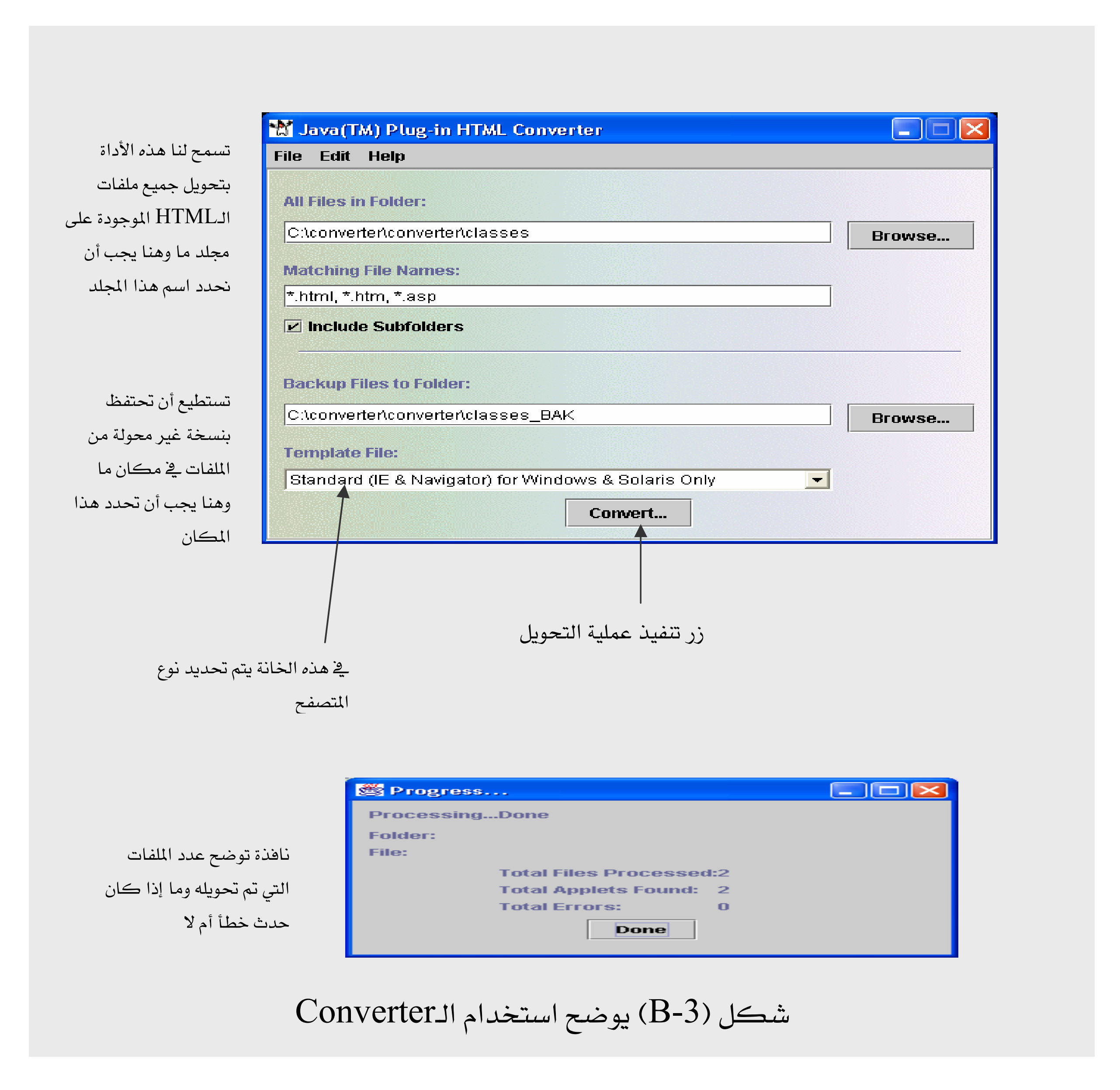
عرض الأبلت على المتصفحات التي لا تدعم الجافا

لعرض أي أبلت على متصفح لا يدعم الجافا نستخدم ما يسمى بـ

Java plug-in HTML converter

وهي أداة تم تطويرها من قبل شركة صن وتستخدم لتحويل ملف HTML المحمل عليه الأبلت إلى ملف آخر بنفس الاسم الامتداد يمكن أن يعرض من خلال المتصفحات التي لا تدعم الجافا.

هذه الأداة موجودة مجانا على موقع الشركة على الإنترنت، بمجرد عمل تحميل لها وتركيبها على الكمبيوتر يمكن أن نشغلها من خلال الملف HTMLConverter.bat ويتم تنفيذ هذا الملف من على محث الدوس طريقة التشغيل. انظر الشكل (B-3):



شكل (B-3) يوضح استخدام الـConverter

References

أولا : المراجع العربية :

١ - **سويلم محمد نبهان**

تحليل وتصميم نظم المعلومات

المكتبة الأكاديمية – القاهرة - ١٩٩٦م

٢ - **علي على يوسف**

تحليل وتصميم نظم المعلومات

خوارزم – القاهرة – فبراير ١٩٩٨

٣ - **د. عوض منصور & د. محمود نحاس**

برمجة باسكال وتيربو باسكال لطلبة الهندسة والعلوم

شبكة الكمبيوتلر الشخصي - مؤسسة الجاسم للإلكترونيات، ١٩٨٧م

ثانيا : المراجع الأجنبية :

١ - Wilson, Thomas C and Shortt Joseph, " Pascal from begin to end",

٢ - Deitel and Deitel , " Java How to Program", Prentice Hall, 2002

٣ - Liang Y. Daniel, " Introduction to Java Programming", Que E&T, 1999

الموضوع **الصفحة**

تقدر المؤسسة العامة للتعليم الفني والتدريب المهني الدعم

المالي المقدم من شركة بي آيه إي سيستمز (العمليات) المحدودة

GOTEVOT appreciates the financial support provided by BAE SYSTEMS

BAE SYSTEMS